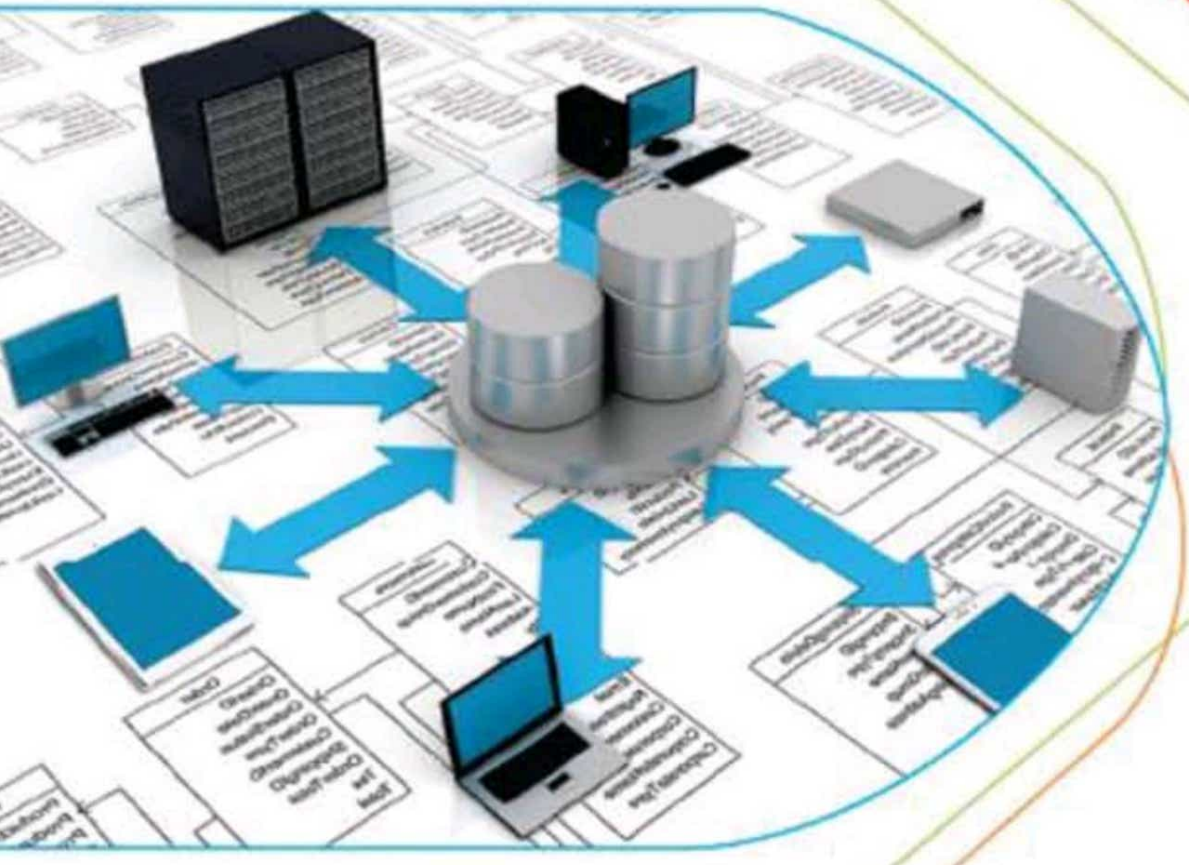
F.Y.B.C.A. SCIENCE : SEMESTER-II

NEW SYLLABUS
CBCS PATTERN

DATABASE MANAGEMENT SYSTEMS-I

ABHIJEET D. MANKAR



NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

SPPU New Syllabus

A Book Of

DATABASE MANAGEMENT SYSTEMS-I

For B.C.A.(Science) : Semester - II

[Course Code BCA124 : Credit - 04]

CBCS Pattern

As Per Revised Syllabus, Effective from June 2019

ABHIJEET D. MANKAR

M.C.S., SET
Assistant Professor,
Department of Computer Science,
Tuljaram Chaturchand College of Arts,
Science and Commerce (Autonomous),
BARAMATI.

Price ₹ 160.00

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

N5114

DATABASE MANAGEMENT SYSTEMS-I **ISBN 978-93-89686-25-8**

First Edition : **November 2019**

Published By :
NIRALI PRAKASHAN
Abhyudaya Pragati, 1312, Shivaji Nagar,
Off J.M. Road, Pune – 411005
Tel - (020) 25512336/37/39, Fax - (020) 25511379
Email : niralipune@pragationline.com

Polyplate

Printed By :
YOGIRAJ PRINTERS AND BINDERS
Survey No. 10/1A, Ghule Industrial Estate
Nanded Gaon Road
Nanded, Pune - 411041
Mobile No. 9404233041/9850046517

➢ DISTRIBUTION CENTRES

PUNE

Nirali Prakashan : 119, Budhwar Peth, Jogeshwari Mandir Lane, Pune 411002, Maharashtra
(For orders within Pune) Tel : (020) 2445 2044; Mobile : 9657703145
Email : niralilocal@pragationline.com

Nirali Prakashan : S. No. 28/27, Dhayari, Near Asian College Pune 411041
(For orders outside Pune) Tel : (020) 24690204; Mobile : 9657703143
Email : bookorder@pragationline.com

MUMBAI

Nirali Prakashan : 385, S.V.P. Road, Rasdhara Co-op. Hsg. Society Ltd.,
Girgaum, Mumbai 400004, Maharashtra; Mobile : 9320129587
Tel : (022) 2385 6339 / 2386 9976, Fax : (022) 2386 9976
Email : niralimumbai@pragationline.com

➢ DISTRIBUTION BRANCHES

JALGAON

Nirali Prakashan : 34, V. V. Golani Market, Navi Peth, Jalgaon 425001, Maharashtra,
Tel : (0257) 222 0395, Mob : 94234 91860; Email : niralijalgaon@pragationline.com

KOLHAPUR

Nirali Prakashan : New Mahadvar Road, Kedar Plaza, 1st Floor Opp. IDBI Bank, Kolhapur 416 012
Maharashtra. Mob : 9850046155; Email : niralikolhapur@pragationline.com

NAGPUR

Nirali Prakashan : Above Maratha Mandir, Shop No. 3, First Floor,
Rani Jhanshi Square, Sitabuldi, Nagpur 440012, Maharashtra
Tel : (0712) 254 7129; Email : niralinagpur@pragationline.com

DELHI

Nirali Prakashan : 4593/15, Basement, Agarwal Lane, Ansari Road, Daryaganj
Near Times of India Building, New Delhi 110002 Mob : 08505972553
Email : niralidelhi@pragationline.com

BENGALURU

Nirali Prakashan : Maitri Ground Floor, Jaya Apartments, No. 99, 6th Cross, 6th Main,
Malleswaram, Bengaluru 560003, Karnataka; Mob : 9449043034
Email: niralibangalore@pragationline.com

Other Branches : Hyderabad, Chennai

Preface ...

We take an opportunity to present this Text Book on **"Database Management System-I"** to the students of Second Semester, B.C.A. Science. The objective of this book is to present the subject matter in a most concise, compact, to the point and lucid manner.

This book aims at clarifying and explaining the fundamental concepts of designing and implementing databases, which is helpful throughout the life time as a student and professional.

Extensive exercises and numerous case studies are provided at the end of each chapter, which helps to develop the skills of database application design and implementation. Without normalization database is of no use. This book clears the concept and enhance the ability of designing normalized database.

We are very much thankful to our Publisher **Shri. Dineshbhai Furia** and **Shri. Jignesh Furia** for publishing this book in minimum possible time.

Any suggestions for improving the content shall be gratefully received and highly appreciated.

Author

Syllabus ...

Unit I	**File Organization**	**08 Hrs**
	Introduction, Physical / logical files, Record organization (fixed, variable length)	
	Types of file organization(heap, sorted, indexed, hashed)	
Unit II	**Introduction of DBMS**	**08 Hrs**
	Overview, File system Vs. DBMS, Describing & storing data (Data models - relational, hierarchical, network), Levels of abstraction, Data independence, Structure of DBMS, Users of DBMS, Advantages of DBMS	
Unit III	**Conceptual Design (E-R model)**	**08 Hrs**
	Overview of DB design	
	ER data model (entities, attributes, entity sets, relations, relationship sets)	
	Additional constraints (key constraints, participation constraints, weak entities, aggregation / generalization), Case studies	
Unit IV	**Structure of Relational Databases**	**08 Hrs**
	Concepts of a table, a row, a relation, a tuple and a key in a relational database	
	Conversion of ER to Relational model	
	Integrity constraints (primary key, referential integrity, Null constraint, unique constraint, check constraint)	
Unit V	**SQL**	**08 Hrs**
	Introduction, DDL commands (create, drop, alter) with examples, Basic structure of SQL query, Set operations, Aggregate functions, Null values, Nested Sub-queries, Modifications to Database (insert, delete, update), SQL mechanisms for joining relations (inner joins, outer joins and their types) Examples on SQL (case studies)	
Unit VI	**Relational Database Design**	**08 Hrs**
	Pitfalls in Relational-Database Design (undesirable properties of a RDB design like; repetition, inability to represent certain information)	
	Functional dependencies (Basic concepts, Closure of set of functional dependencies, Closure of an Attribute set)	
	Concept of a Super Key and a primary key (Algorithm to derive a Primary Key for a !relation)	
	Concept of Decomposition, Desirable Properties of Decomposition (Lossless join and Dependency presE)rvation)	
	Concept of Normalization - Normal forms (only definitions) 1NF, 2NF, 3NF, BCNF. Examples on Normalization.	

Contents ...

1.	File Organization	1.1 – 1.19
2.	Introduction to DBMS	2.1 – 2.18
3.	Conceptual Design (E-R Model)	3.1 – 3.28
4.	Structure of Relational Databases	4.1 – 4.17
5.	SQL	5.1 – 5.44
6.	Relational Database Design	6.1 – 6.30
*	Bibliography	B.1 – B.1

1... File Organization

Objectives...

- To know about the fundamental concepts of file and file organization.
- To know about file operations.
- To get familiar with record organization.
- To have knowledge about various types of file organization.
- To understand the use of different file organizations and to find the differences among them.

1.1 INTRODUCTION (S-17, W-17)

- A file in an office maintains the records/details about specific activity. e.g. admission file, examination file etc. An examination file contains data about the examination such as candidate's roll number, subject etc.
- A file is a sequence of related records.
- A file organization is a method which mentions about the physical arrangement of the data of a file on storage medium.
- The elements which make a file are records, fields and characters.
  - **Record:** It is a collection of related fields (columns).
    e.g. In an Employee file, details about an employee like emp_no, designation, joining_date, salary make up a record.
  - **Field:** It is a single character or group of characters representing a small part of data.
    e.g. emp_no is an example of a field (column).
  - **Character:** It is the smallest element in a file. It can be letter (alphabet), number or symbol that is stored by a computer.
- A file having 'n' records and each record having 'm' fields is shown in Fig. 1.1.

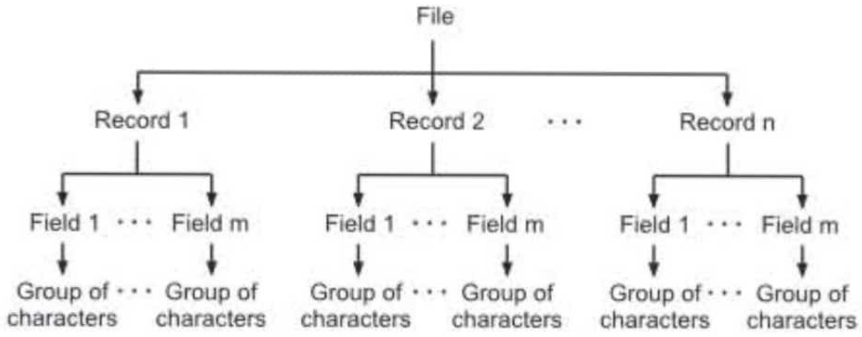


Fig. 1.1: Elements of a File

- Some of the file operations are explained below:
  1. **Open:** This operation opens a file for either reading or writing. The file pointer of the file is set to the beginning of the file.
  2. **Close:** This operation frees the buffer area and memory allocated to a file.
  3. **Seek:** Sometimes we may want to move directly to a specified position in a file. For that purpose, seek operation is used. It needs two arguments: filename, offset.
  4. **Read:** In this operation, the current record from the main memory buffer is taken into user's working area of RAM. It may move the pointer to the next immediate record in the file, which may further need to get the file block containing that record from disk (if it is not present in the buffer).
  5. **Write:** In this operation, user can edit the contents of a file. After editing the file contents, this operation updates the file on secondary storage device to reflect the changes.

1.2 PHYSICAL/LOGICAL FILES (S-17, W-18)

- File consists of records, records consist of fields.
- There are two different ways of viewing files.
  1. **Logical File:** This describes what data items and possible processing operations can be performed on a file.
  2. **Physical File:** This describes how the data is stored or arranged on a storage device and how processing operations are made possible over the stored data.
- An application program instructs operating system to connect to a physical file through a logical file. Hence, a logical file acts as a mediator between an application program and a physical file.

  e.g. In a C language program code,

```
FILE *fpt;
fpt = fopen("stud.txt", "w");
```

  fpt is the logical file name and stud.txt is the physical file name.

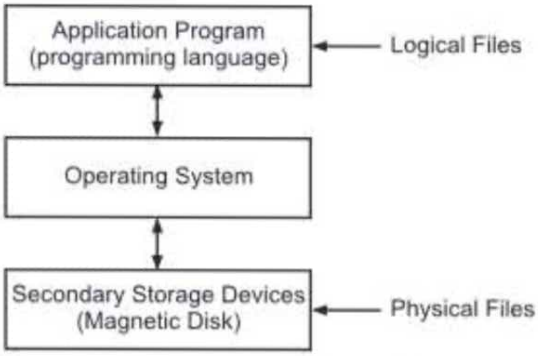


Fig. 1.2: Logical and Physical files

Difference between Physical File and Logical File

Sr. No.	Physical File	Logical File
1.	This describes how the data is stored or arranged on a storage device.	This describes what data items and possible processing operations can be performed on a file.
2.	Physical file can exist without logical file.	Logical file cannot exist without a physical file.
3.	It contains the original data.	It does not contain any data.

1.3 RECORD ORGANIZATION (W-17, S-18)

- A file is a collection of logically related records.
- There are two types of record:
  (i) **Fixed Length Records:** In a file having fixed length records, every record in the file is equal in size (in bytes).
  (ii) **Variable Length Records:** In a file having variable length records, different records in the file do not have equal size (in bytes).

	Emp_id	Emp_name	Desg	Salary
Record 1	101	Krishnamurthy	Manager	60000
Record 2	102	Ahluwalia	Cashier	55000
.				
.				
.				
Record n				

(a) Fixed Length Records

Record 1	101	Krishnamurthy	Manager	60000
Record 2	102	Ahluwalia	Cashier	55000
.				
.				
.				
Record n				

(b) Variable Length Records

Fig. 1.3: Types of Record

1.3.1 Fixed Length Records

- Consider a file of EMPLOYEE records.
- Each record of the file is defined as follows:

```
EMPLOYEE
(emp_id char (10),
emp_name char (20),
emp_addr char (50),
emp_state char (20))
```

- If we assume that each character occupies one byte, then EMPLOYEE record takes 100 bytes.
- Fig. 1.4 shows a record of EMPLOYEE file.

Fig. 1.4: Fixed Length Record

- As every record consists of same number of fields and size of each field is fixed, the system can easily identify the starting position of each field. Also identifying start and end of record is easy and simple.

Advantages:

1. Access to record is fast because record length is fixed.
2. Insertion and deletion operations are easy to implement.

Disadvantages:

1. A lot of memory space is wasted.
2. To fill up the blank space created by deletion of a record. The records following the deleted record are shifted.

1.3.2 Variable Length Records

- If one or more fields of the file record are optional (i.e. may or may not have values for a record), then a file may have variable length records.
- In this type, to predict the exact length of field in advance is difficult. Hence to determine the boundary of each field in a record, we need special separator characters.

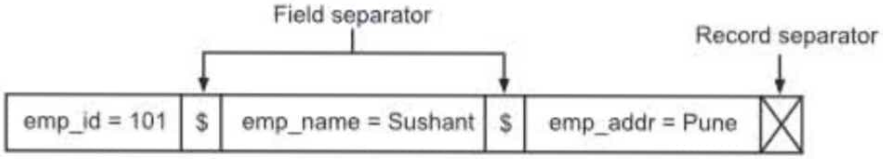


Fig. 1.5: A variable length record with separator characters

- In Fig. 1.5, there are three separator characters, namely =, $ and ☒. '=' is used for separating the name of the field from its actual value.
- In order to represent a file having variable-length records as a fixed length records file, a special null value is stored for optional fields. It is somewhat difficult to guess the exact length of some field values, in case of variable length fields. Special characters or delimiters (e.g. $ or %) are used to indicate termination of variable length fields.
- A record with optional fields can be represented as a sequence of <field-name, field-value> pair. It is shown in Fig. 1.5.
- A repeating field within a record needs two separator characters:
  (i) To separate the repeating values of the field.
  (ii) To show termination of the field.
- For a file having records of heterogeneous types, a record type indicator is used (or added) before every record.
- An array of field offsets technique is also used to organize the variable length records.

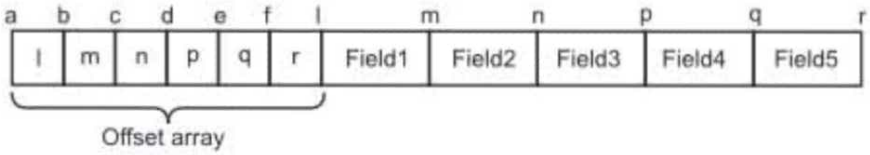


Fig. 1.6: Organization of variable length records using an array of field offsets

- In case of a null value, the pointer to start and end of field is same. An offset indicating the end of record is also stored, which also recognizes the end of the last field. Maintaining an offset array is an extra overhead. But it gives direct access to any field of a record.

Advantages:

1. Variable length records save a lot of memory space.
2. It is flexible.

Disadvantages:

1. Not so easy to determine the start of each record, hence processing the record get slower.
2. Increased overhead to keep record of the sizes of all records.

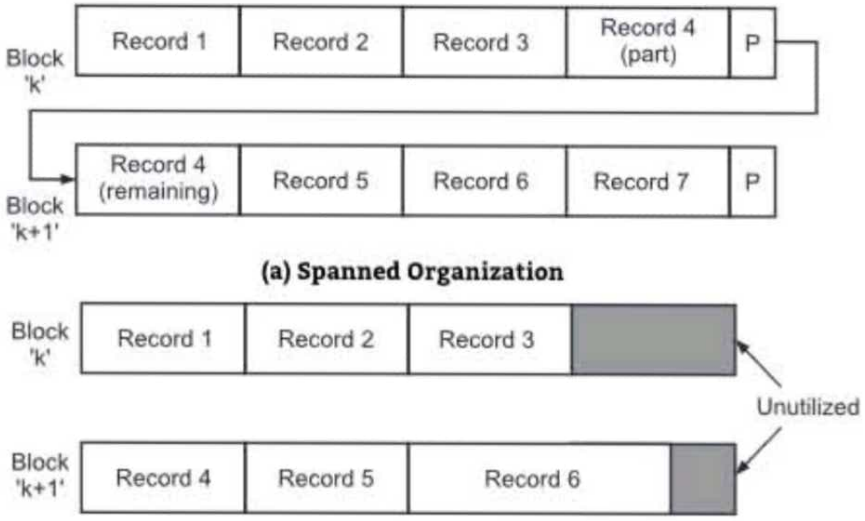
Difference between Fixed Length Record and Variable Length Record:

Sr. No.	Fixed Length Record	Variable Length Record
1.	In a file having fixed length records, every record is of equal size (in bytes).	In a file having variable length records, different records in the file do not have equal size (in bytes).
2.	Access to record is fast because record length is fixed.	Access to record is slow because it is not so easy to determine the start of each record.
3.	A lot of memory space is wasted.	It saves a lot of memory space.
4.	Insertion and deletion operations are easy to implement.	Insertion and deletion operations are not easy as compared to fixed length record.

Spanned and Unspanned Organization:

- The records of a file are assigned to disk blocks using either spanned or unspanned organization.

1. **Spanned Organization:** When, size (block) > size (record), a block can store several records. Some records might cross block boundaries. In such case, a portion of a record is stored in one block and another portion is stored in another block. This organization is called Spanned Organization.
2. **Unspanned Organization:** In this type, a record is stored on single block and is not allowed to be stored on two different blocks in splitted manner. It is suitable for fixed-length records. It also makes processing of records easier. The choice between spanned and unspanned organization is dependent on two factors, storage space and record processing. Spanned organization saves the storage space. On the other hand, unspanned organization makes the record processing easier.



(a) Spanned Organization

(b) Unspanned Organization

Fig. 1.7: Types of Record Organization

1.4 TYPES OF FILE ORGANIZATION (S-17, S-18)

- A file organization is a method of arranging records in a file (on secondary storage).

1.4.1 Heap File

- Heap file is the simplest type of file organization. In this type, the records are stored in the file in the same order in which they are inserted. The new records are appended to the end of file. A heap file is also known as a pile file or serial file.
- In Fig. 1.8, shows a heap file organization for EMPLOYEE relation. We assume that each block contains at most three records.

EMPLOYEE

	EMP_ID	EMP_NAME	EMP_SAL	EMP_ADDR
Block 1	105			
	103			
	110			
Block 2	102			
	106			
	109			
Block 3	107			
	101			

Fig. 1.8: Heap File Organization

- Operations on heap file:
  1. **Insertion:** Inserting (adding) a new record to an existing heap file is easy. The new record is added after the last record of the file.
  2. **Search:** Searching for a record requires sequential search. In this organization we need to scan the entire file. This operation is not efficient if the file is large.
  3. **Deletion:** To delete an existing record, a program must find the block which contains that record. Then the record is deleted from that block. Finally the block present in main memory is over -written back to disk. Deletion of record in this manner leads to wastage of storage space.
  4. **Update:** To update a record, we need to find the block which contains the record to be updated. Then the block is copied into buffer, changes are done to the record and the block (in main memory) is rewritten back to disk.

Advantages:

1. This is simple file organization method.
2. Space is fully utilized.
3. Insertion of new records is fast.

Disadvantages:

1. Searching record is a slow operation.
2. Cost of updating records is high.
3. Deletion of many records result in wastage of storage space.

1.4.2 Sorted File (S-17)

- Often it is required to process the records of the file in the sorted order based on the value of one of its fields, known as Search Key.
- A search key is used to arrange all records in sequential order. A search key can be one attribute or a set of attributes.
- Fig. 1.9 shows a Sorted File Organization for EMPLOYEE relation.

EMPLOYEE

	EMP_ID	EMP_NAME	EMP_SAL	EMP_ADDR
Block 1	101			
	102			
	103			
Block 2	105			
	106			
	107			
Block 3	109			
	110			

Fig. 1.9: Sorted File Organization (according to EMP_ID)

- Operations on sorted file:
  1. **Search:** To search a record, linear search is performed.
  2. **Insertion and Deletion:** Insertion and deletion of records are expensive because the records in the file should remain physically ordered based on search key.
  3. **Update:** Updating a field value in a record requires first to search the record. If the field, that defines the physical ordering of records, is changed then the position of the record in the file in changed.

Advantages:

1. Access to the next record is easier.
2. This file organization is simple to understand.
3. Improvement in query execution time as compared to similar query executed on a heap file.
4. Every record is stored on the next available space, thereby leaving no gaps between the records.

Disadvantages:

1. Insertion and deletion are expensive operations.
2. Deletion creates wastage of storage space and requires shifting (reorganization) of records.

Difference between Heap File and Sorted File:

Sr. No.	Heap File	Sorted File
1.	Searching record is a slow operation as there is no order among the records.	Searching record is faster as there is order among records.
2.	Insertion of new record is fast.	Insertion is expensive operation.
3.	Query execution time is slower as compared to sorted file.	Improvement in query execution time as compared to similar query executed on a heap file.
4.	Heap file is useful when data is collected prior to processing.	Sorted files are useful in case of batch oriented processing.

1.4.3 Indexed File (S-17, W-17)

- The retrieval of a record from a sorted file requires access to nearly half of the records from the file. This is rather time consuming for files having large number of records.
- A sequential file that is indexed is called as Indexed Sequential File. In indexed sequential access method, records are stored on disk-sequentially by key value, but indexes are also maintained to allow direct retrieval based on a key value.

- For example, we are looking for 'theme' word in English language dictionary. There is a thumb index provided in the dictionary. We access the appropriate thumb index 'T'. Then we find the word sequentially.
- The files are organized in following manner:
  1. A separate file stores the data in the order of primary key values. (i.e. data file)
  2. Another file is maintained to store index entries. It contains two fields: key value, pointer to the data record.
  3. A record in the index file contains a key value and a pointer to the appropriate data record. The pointer points to the first entry in the range of data records.

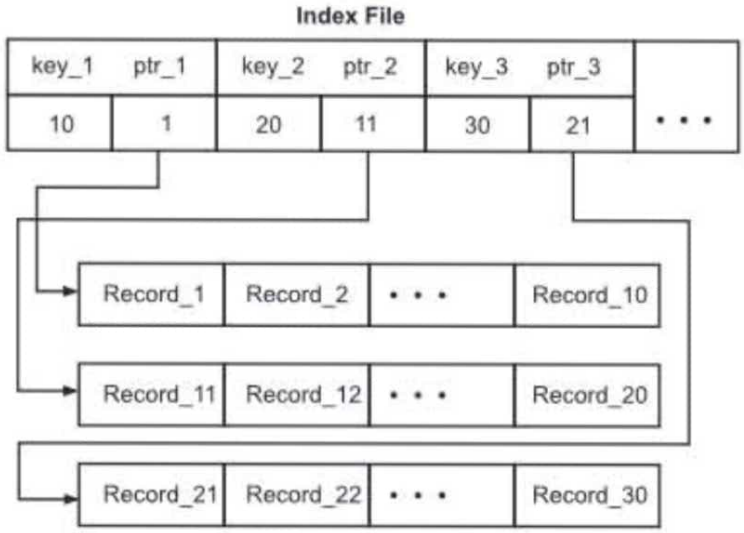


Fig. 1.10: Index File Organization

- In Fig. 1.10, the first key value is 10, which is the highest key value in the first data block of 1 to 10. The pointer of this index entry refers to the start of the range. i.e. 1.

Advantages:

1. In this organization, the desired record can be immediately accessed and it is unnecessary to read the entire file.
2. It gives faster access as compared to sequential file.
3. Primary and secondary index can be used to search the data.

Disadvantages:

1. The indexed files have to be reorganized periodically to discard deleted records and improve performance.
2. There is an extra overhead of maintaining additional data structures. These data structures can utilize more disk space for long key values.
3. Here the back-up should be taken regularly.

Difference between Heap File and Indexed File:

Sr. No.	Heap File	Indexed File
1.	Searching record is a slow operation as there is no order among the records.	In this, the desired record can be immediately accessed and it is unnecessary to read the entire file.
2.	There is no extra overhead.	There is an extra overhead of maintaining additional data structures.
3.	Heap file is not reorganized periodically.	The indexed files are reorganized periodically to improve performance.
4.	Heap file is useful when data is collected prior to processing.	Indexed sequential file is used in commercial data processing applications.

Indexing:

- Indexing is a data structure technique to efficiently retrieve records from database files based on some attributes on which the indexing has been done.
- Indexing can be one of the following types:
  1. **Primary Index:** If index is built on ordering key field of the file, it is called Primary Index. In this type, every record has unique value for that field.
  2. **Secondary Index:** If index is built on non-ordering field of the file, it is called Secondary Index.
  3. **Clustering Index:** If ordering field is not a key field, then the index is Clustering Index. In this type, several records in the file can have same value for the ordering field.

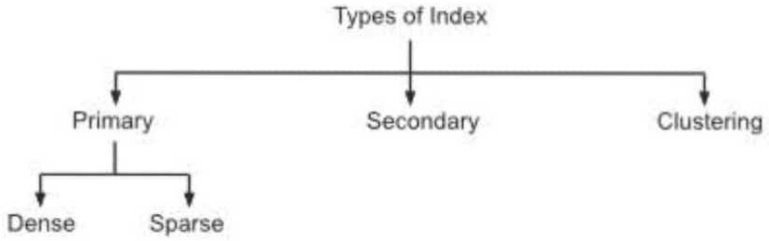


Fig. 1.11: Types of Index

i. Dense Index:

- There is an index record for every search key value in the database. Searching speed is fast, but it requires more storage space.

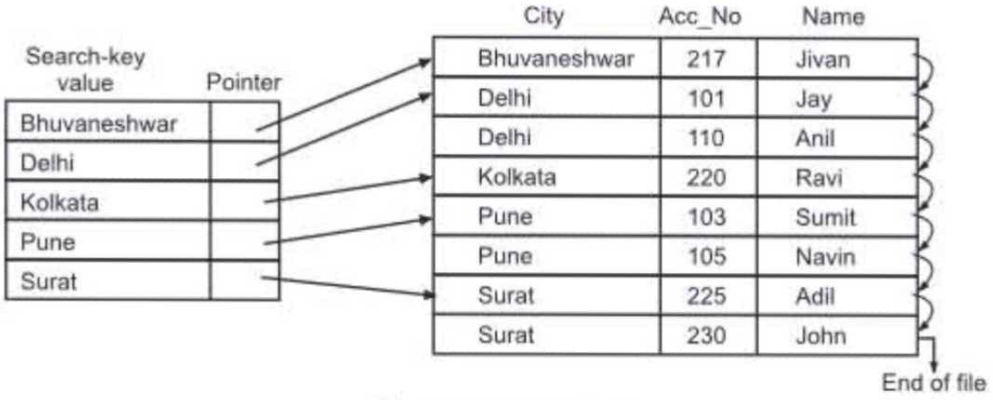


Fig. 1.12: Dense Index

ii. Sparse Index

- In this type of index, index records are created for only some of the records. To search a record, we find the index record with the largest search-key value that is less than or equal to the search-key value we are looking for. We proceed by that index record and follow the pointers until the desired record is found.

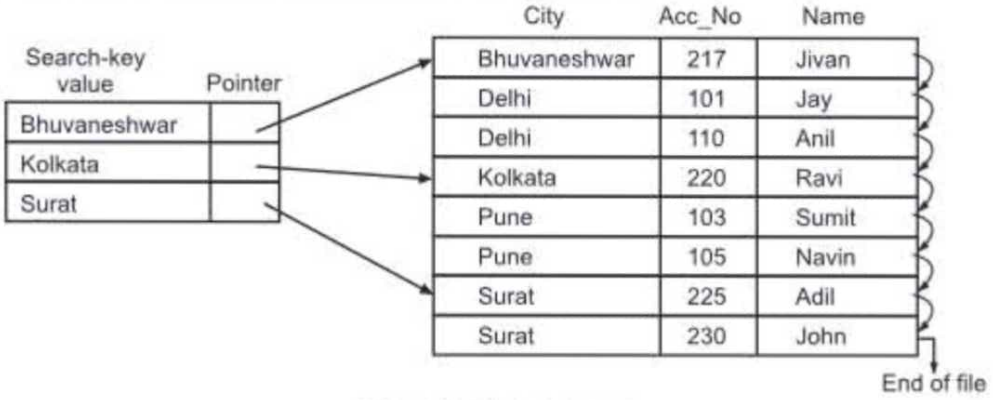


Fig. 1.13: Sparse Index

Difference between Dense Index and Sparse Index:

Sr. No.	Dense Index	Sparse Index
1.	In dense index, there is an index record for every search key value.	In sparse index, index records are only for some of the search key values.
2.	As there is an index entry for every search key value, there is direct access to a record.	In sparse index, we must search within block for accessing a record.

Cond..

3.	Dense index is updated each time if the order of records changes.	Sparse index is updated only if first record in block is changed.
	Dense index requires more storage space.	Sparse index saves storage space.

1.4.4 Hashed File

- A hashed file uses a hash function applied to a particular field to find the position of a record on disk. This organization is also known as Random Access File Organization. There are two types of Hashing:
  (a) **Internal Hashing:**
  - For internal files, hashing is typically implemented through the use of an array of records. We assume that array index range is from 0 to N – 1. We have N slots. We choose a hash function h(K) = K mod N, which returns the remainder after division. This value is then used for the record address.
  - A collision occurs when the hash field value of a new record that is being inserted hashes to an address that already contains a different record. To find another position is called **Collision Resolution**.
  - **Methods of Collision Resolution:**
    (i) **Open addressing:** Starting from the filled position specified by the hash address, the program verifies the next positions in sequence until a vacant position is found.
    (ii) **Chaining:** A collision is resolved by placing the new record in a vacant overflow location and adjusting the pointer of the filled hash address location to the address of that overflow location.
    (iii) **Multiple hashing:** The program make use of second hash function, if the first one results in collision problem. In case of collision repetition, the program make use of a third hash function and then make use of open addressing or just uses open addressing.

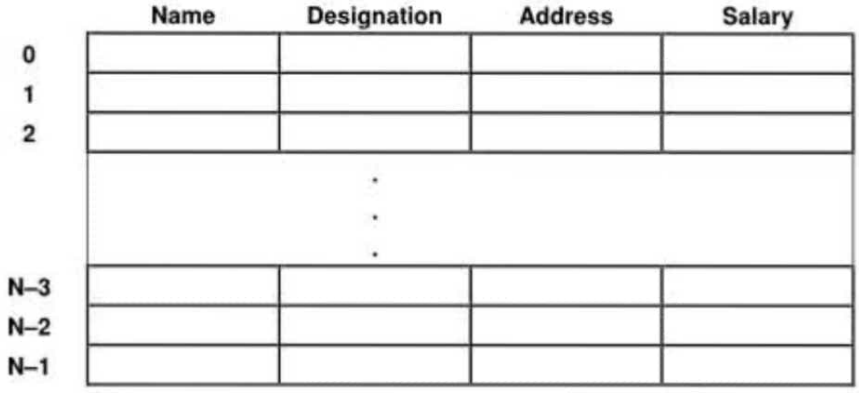


Fig. 1.14: Internal hashing using array of 'N' position

(b) External Hashing:

- Hashing for disk files is known as External Hashing. The address space of disk is divided into buckets, each one of these hold multiple records. In hashed file organization, the records are grouped together into buckets. A bucket is either one disk block or a bunch of contiguous blocks. The hashing function maps a key into a relative bucket number, rather than assigning an absolute block address to the bucket. A table kept in the file header transforms the bucket number into the corresponding disk block address. It is shown in Fig. 1.15.

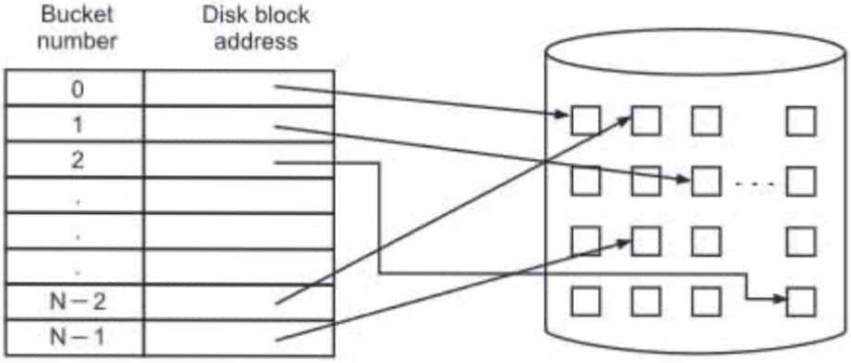


Fig. 1.15: Matching of bucket numbers with disk block addresses

- In a hash file, the data is dispersed throughout the disk in a random manner. The processing of a hash file is dependent on how the search key set for the records is transformed into the addresses of hard disk to find the desired records. In most cases, the hash field is also a key field of the file, which is called hash key. There is a function h called as hash function or randomizing function which is applied to hash field value of a record. This function gives the address of the disk block in which the address is stored. The search for a record within the block can be carried out in main memory buffer.

Advantages:

1. The data is processed immediately.
2. Since the records can be accessed directly, the file updating is easy.
3. Record is accessed directly and quickly.

Disadvantages:

1. If we are searching for range of data, then this method is not suitable.
2. This method is efficient only when the search is performed on hash field.
3. If hashed column is frequently updated, it results in movement of data between buckets which actually affects the system performance.

Difference between Heap File and Hashed File

Sr. No.	Heap File	Hashed File
1.	Heap file is known as serial file or pile file.	Hashed file is known as random access file organization.
2.	Searching record is a slow operation as there is no order among the records.	Searching is fast as compared to heap file. This method is efficient only when the search is performed on hash field.
3.	Insertion of new record is fast. New records are appended to the file.	For inserting a new record, we need to apply a hash function h(K) which return the bucket address for the new record.
4.	There is no overhead.	If hashed column is frequently updated, it results in movement of data between buckets which actually affects the system performance.

Summary

- File organization refers to the way of arranging the data in a file.
- Logical file is related with the 'what' aspect and physical file is related with the 'how' aspect of data.
- The operations that can be performed on a file are open, close, seek, read, write.
- A file is a collection of logically related records.
- There are two different record types: fixed-length record, variable-length record.
- In a file having fixed-length record, every record is of the same size (in bytes).
- In a file having variable-length record, different records in the file have different size (in bytes).
- Access to a record is fast in case of file having fixed-length records, but a lot of memory space is wasted.
- Variable-length record saves a lot of memory space, but processing of the record gets slower.
- Spanned organization saves storage space. On the other hand, unspanned organization makes the record processing easier.
- In heap file new records are appended to the end of file. It is also called pile file or serial file.
- In case of heap file: search is slower, insertion is faster.

- Heap file is useful when data is collected prior to processing.
- In case of Sorted file, insertion and deletion are expensive operations.
- Sorted files are useful in case of batch-oriented processing.
- A sequential file that is indexed is known as Indexed Sequential File.
- Dense Index has an index record for every search-key value in the database.
- Sparse Index has index records for some of the search-key values in the database.
- Indexed sequential file is used in commercial data processing applications.
- Hash file is also known as Random Access File Organization.

Check Your Understanding

Q.1. Multiple Choice Questions:

1. A collection of related fields is called .....
   (a) file (b) record
   (c) byte (d) column
2. ..... describes what data items and possible processing operations can be performed on a file.
   (a) Logical file (b) Physical file
   (c) both (d) none of these
3. In ..... record memory space is saved.
   (a) fixed-length (b) variable-length
   (c) mixed length (d) none of these
4. In ..... file records are stored in the file in the same order in which they are inserted.
   (a) hash (b) sorted
   (c) indexed-sequential (d) heap
5. A sequential file that is indexed is called as ..... file.
   (a) heap (b) sorted
   (c) hash (d) indexed-sequential
6. In ..... index, index records are created for only some of the records.
   (a) sparse (b) dense
   (c) both (d) none of the above
7. Which of the following is not a collision resolution method ?
   (a) Open addressing (b) Chaining
   (c) Multiple hashing (d) External hashing

8. A storage unit that can store one or more records in hash file organization is called as .....
   (a) disk pages (b) buckets
   (c) blocks (d) nodes
9. A clustering index is defined on the fields which are of type .....
   (a) non-key and ordering (b) key and ordering
   (c) none-key and non-ordering (d) key and non-ordering

Ans.: (1) - b (2) - a (3) - b (4) - d
(5) - d (6) - a (7) - d (8) - b
(9) - a

Q. 2. State True/False.

1. Hashing is a method of determining the physical location of a record.
2. When deletion of many records is done in a heap file, it results in wastage of storage space.
3. Searching a record is faster in heap file.
4. The indexed file have to be reorganized periodically to discard deleted records.
5. Primary Index can be built on non-ordering field.
6. In dense index, index records are created for only some of the search-key values.
7. A bucket can hold multiple records.
8. Chaining is a collision resolution method.
9. File is a set of logically related records.

Ans.: (1) True (2) True (3) False (4) True
(5) False (6) False (7) True (8) True
(9) True

Practice Questions

Q.1. Answer the following questions in brief.

1. What are the different types of records ?
2. How are records and files related ?
3. What is a sorted file organization ?
4. What is indexed-sequential file organization ?
5. State the difference between logical and physical file.
6. What is fixed-length record ?
7. What is the drawback of variable-length record ?

Q.2. Answer the following questions.

1. Compare fixed-length record and variable-length record.
2. Explain heap file organization.
3. Explain the difference between dense and sparse index.
4. Explain different types of indexes.
5. Write advantages and disadvantages of sorted file.
6. Compare between sorted file and heap file.
7. Compare between sorted file and indexed-sequential file.
8. What are collision resolution methods ?
9. What is external hashing ?

Q.3. Define the terms.

(a) Field
(b) File
(c) Record
(d) Collision
(e) Clustering Index
(f) Hash function
(g) Logical file
(h) Physical file
(i) Chaining
(j) Open addressing

Previous Exams Questions

Summer 2017

1. The files of randomly ordered records are called as ..... files. **[1 M]**

   (i) index
   (ii) heap
   (iii) both (i) and (ii)
   (iv) None of the above

Ans.: Refer to section 1.4.

2. 'A sorted file is best if range selection is designed'. State True or False. Justify your answer. **[1 M]**

Ans.: Refer to section 1.4.2.

3. What is Indexed File Organization. **[4 M]**

Ans.: Refer to section 1.4.3.

4. Explain concept of overflow pages in ISAM. **[4 M]**

Ans.: Refer to section 1.4.3.

5. Define file. Discuss physical and logical file in detail. **[4 M]**

Ans.: Refer to section 1.1 and 1.2.

Winter 2017

1. A set of logically related record forms a ..... [1 M]
   (a) database
   (b) file
   (c) record
   (d) none of the above

Ans.: Refer to section 1.1.

2. Define Record Type. [1 M]

Ans.: Refer to section 1.3.

3. What is ISAM. [1 M]

Ans.: Refer to section 1.4.3.

4. What do you mean by fixed and variable length record? Explain with example. [4 M]

Ans.: Refer to section 1.3.

5. What do you mean by index organization ? How is it implemented using dense index and sparse index ? [4 M]

Ans.: Refer to section 1.4.3.

Summer 2018

1. Which of the following is not the type of file organization ? [1 M]
   (i) heap
   (ii) sort
   (iii) hash
   (iv) sparse

Ans.: Refer to section 1.4.

2. What do you mean by physical and logical file ? [1 M]

Ans.: Refer to section 1.2.

3. Define variable-length record. [1 M]

Ans.: Refer to section 1.3.

❖❖❖

2... Introduction of DBMS

Objectives...

- To understand basic concept of data.
- To have knowledge about different data models.
- To know about DBMS and its applications and users.
- To understand the concept of data independence.
- To know the advantages of DBMS.
- To understand the difference between File Processing System and DBMS.

2.1 OVERVIEW (S-18)

- Today, databases have become an integral part of our life. In our day-to-day life, we come across situations that involve some interaction with a database. For example, we purchase some products online. In this activity, the website through which we are purchasing products is definitely accessing a database which stores the details about the product such as stock in hand, rate, description etc.
- A database is a collection of related data that is stored in a standardized format, sharable among multiple users.
- A Database Management System (DBMS) is a software system or program that allows access to data stored in database.
- The database and the DBMS software combined together known as Database System. In other words, a database system is an integrated collection of related files, along with the details of the interpretation of the data.
- **Data:** Data is defined as a collection of raw facts that have implicit meaning.
- **Information:** Information is obtained after processing the data.

Difference between Data and Information

Data	Information
1. Data is derived from the latin word 'datum' which means 'to give'.	1. Information is derived from the latin word 'informare' which means 'to instruct'.

Contd...

2. Data is in raw format.	2. After processing data we obtain information.
3. Data can be simple and at the same time unorganized.	3. Information is a set of data which is processed in a meaningful way.
4. Data may or may not be meaningful.	4. Information does not contain meaningless detail.
5. Data doesn't play any role in decision making.	5. Information plays an important role in decision making.
6. Example: Sales done by a supermarket on a day.	6. Example: Regionwise sales report. It gives an idea about the most profitable region.

- **Field (data item):** A filed is a single unit of data in a stored record. Each field has a certain data type.
  For example: In the Student file, roll_number is a field.
- **Record:** A collection of different fields that provide information on an entity is called a record.
  For example: 11, Rohit, Manager is a record of emp_id = 11, name = Rohit and designation = Manager.

2.2 FILE SYSTEM VS DBMS (W-17)

- In a file system, the records are stored in separate files. Each file is called a flat file. For accessing the data from these files, various programs are written. As system became more complex, file processing system loses its flexibility.
- The file system has limitations which are mentioned as follows:
  1. **Data Redundancy:** It means same information is unnecessarily duplicated in several files.
  2. **Separated and Isolated data:** Data are scattered in various files and the files may be in different format. Writing a new application program to fetch data is difficult.
  3. **Difficulty in accessing data:** If the format of a certain record was changed, the code must be immediately updated, otherwise system provides incorrect data.
  4. **Integrity problems:** The data values may need to satisfy some integrity constraints.
  5. **Concurrent Access Anomalies:** In file processing system, once a file is opened by a user, the access to this file is denied for another user, till the first user closes the file. Sharing a file by various users at the same time is not possible.
  6. **Security Problems:** Enforcing security constraints in file processing system is very difficult as the application programs are added to the system in an ad-hoc manner.

Components of database system:

1. **User:** These are the people interacting with the database system. There are four types of users: Database Designers, Application Programmers, Sophisticated Users, End Users.
2. **Data:** Data is one of the important factor of database. The data in the database are both shared and integrated.
3. **Hardware:** It consists of the secondary storage devices like disk, where the database resides.
4. **DBMS:** It is a layer or interface of software that is present in between the physical database and the users. All requests from the users to access the database are handled by the DBMS.
5. **Application Program:** It helps the users to interact with the database by means of query languages.

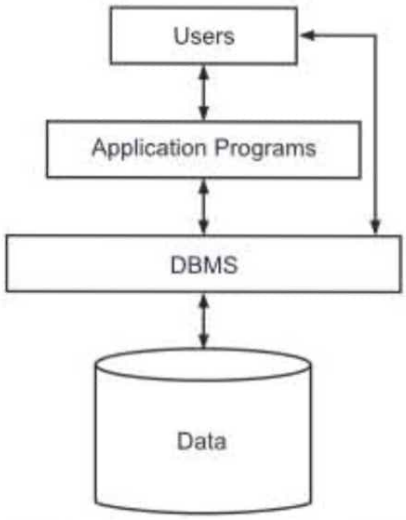


Fig. 2.1: Components of Database System

Comparison between File System and DBMS:

File System	DBMS
1. Data is isolated.	1. Data is integrated.
2. Doesn't provide security.	2. Provides better security by assigning user name and password.
3. Concurrency control is not possible.	3. Concurrency control is possible.
4. Transaction concept is not present.	4. Transaction concept is present.
5. They are relatively cheap.	5. They are relatively expensive.

Contd...

6. It needs very little preliminary design.	6. It needs vast preliminary design.
7. They are often single user oriented.	7. They are multiple user oriented.
8. Data access is difficult.	8. Data access is efficient.
9. Examples: Small system like a C++, COBOL program.	9. Examples: Large System like Oracle, Postgres etc.

2.3 DESCRIBING AND STORING DATA

- Data stored in DBMS describe real world entities and represent relationship between them. Data can be described through different data models.

2.3.1 Data Models (S-17, W-17)

- A model is an abstraction process that hides the details which are not required while highlighting details required to the applications at hand.
- A data model is a collection of concepts that can be used to describe the structure of a database.
- There are three types of data models available.
  1. Object based Logical Model
  2. Record based Logical Model
  3. Physical Data Model

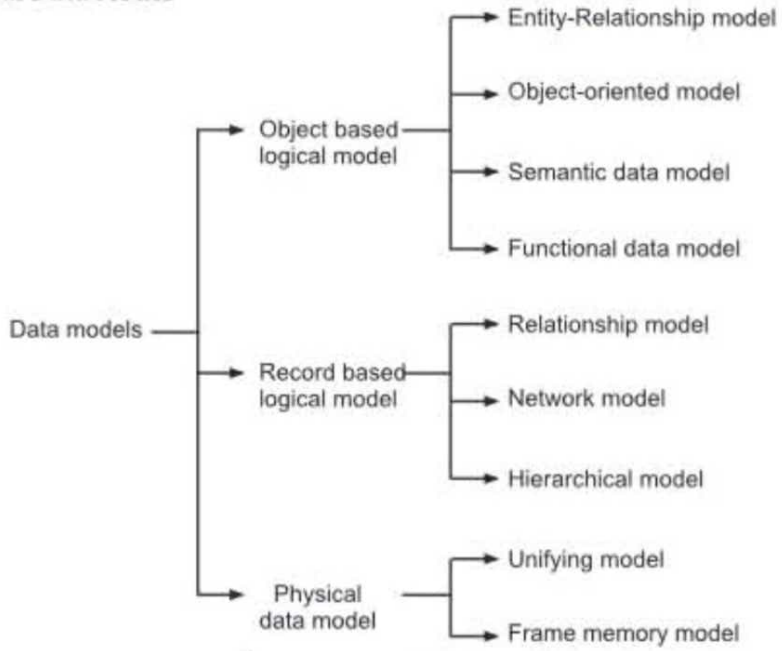


Fig. 2.2: Types of Data Model

2.3.2 Record based Logical Models

- These data models are used in describing data at the logical and view levels.
- These are used to describe the logical database structure in terms of records.
- There are three types of models in this category:
  1. Relational Model
  2. Network Model
  3. Hierarchical Model

1. Relational Model: **(W-17)**

- The Relational Model was introduced by Dr. E.F. Codd in 1970. The Relational Model represents data in the form of two-dimension tables called as relations. Each table consists of multiple columns and each column is identified by a unique name.
- A data value is stored in the intersection of a row and column. Each column has a domain, i.e. the set of acceptable values that may appear in that column.
- For example, in a database Student table and Subject table are the two tables available. The relationship between these two tables is shown by a third table, where one field or column name from both the table is taken together. The new table Examination_Info is formed.

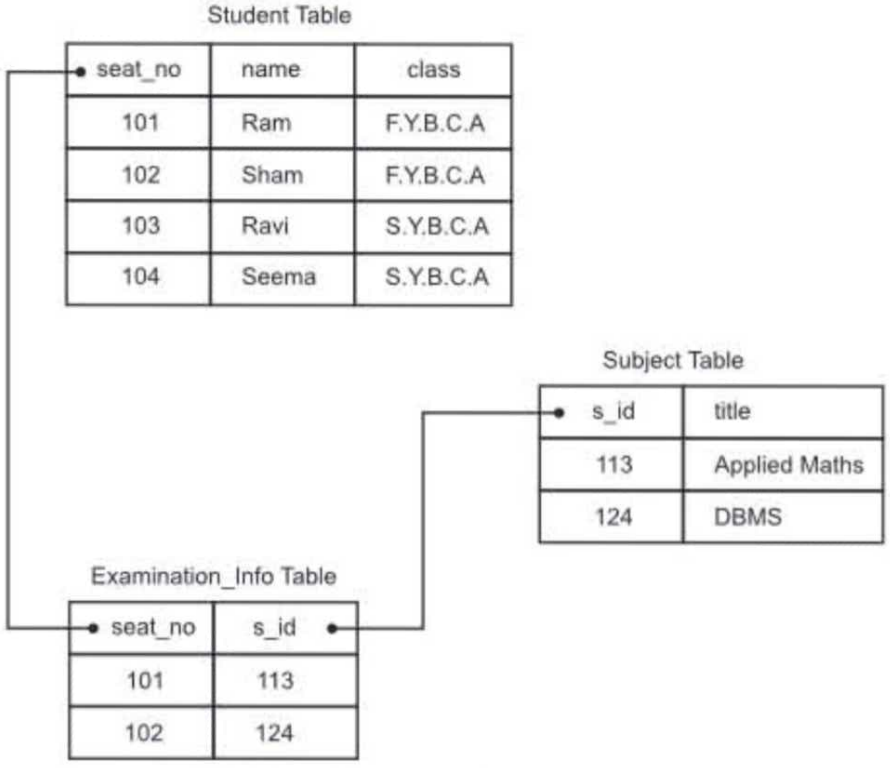


Fig. 2.3: Relational Model Representation

2. Network Model: **(S-17)**

- In late 1960s, Network Model was formalized.
- In Network Model, data is represented by collection of records and the relationship among the data is represented by links. The records are organized as a collection of arbitrary graphs. Fig. 2.4 shows Network Model.
- For example, in the Student table extra field or column is considered which consists of the address where the Subject table information is stored. Each link information represents the related record address.

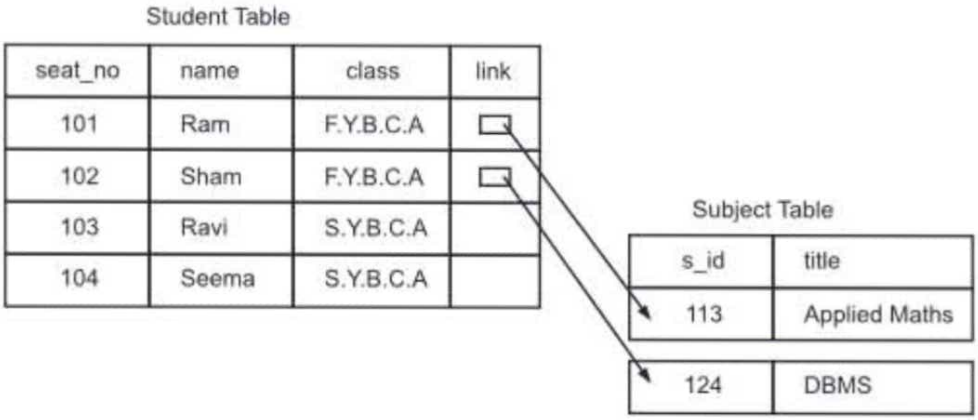


Fig. 2.4: Network Model Representation

3. Hierarchical Model:

- This model is developed by IBM in 1960s.
- In this model, data is organized into a tree-like structure in which each child node can have only one parent node.
- At the top of the hierarchy there is a single node called as root.
- Data in this model is similar to network model. Only change is that the records are organized as a collection of tree.

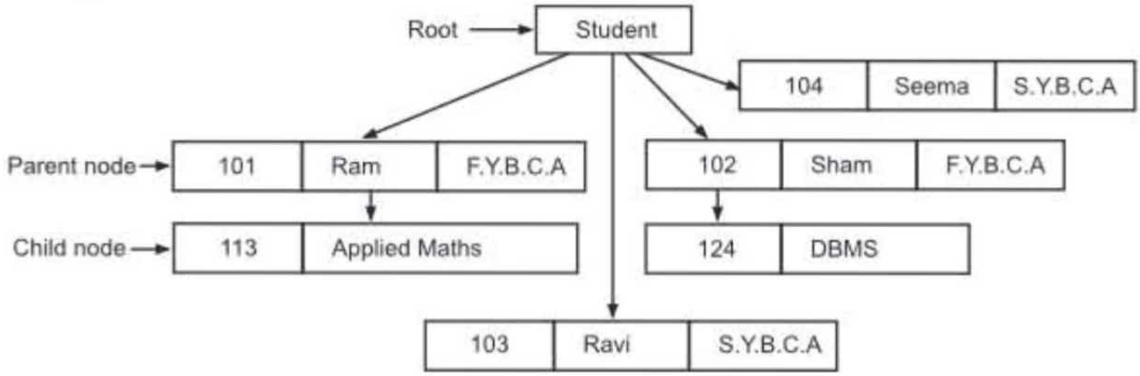


Fig. 2.5: Hierarchical Model Representation

- Comparison between Hierarchical Model, Network Model and Relational Model:

	Hierarchical Model	Network Model	Relational Model
1.	It is based on tree (hierarchy) structure.	It is based on graph (tree of records) structure.	It is based on mathematical concept of relations.
2.	Structural independence is missing.	Structural independence is missing.	It offers structural independence.
3.	Queries are easy to write than in network model but more complicated than the relational model.	Queries are more complicated to write than hierarchical and relational model.	Queries are easy and simple to write than other models.
4.	Data access is navigational.	Data access is navigational.	Data access is non-navigational.
5.	Easy to understand and more efficient than network data model.	Offers more flexibility than the hierarchical model.	Offers simplicity in representing data than network and hierarchical models.
6.	The hierarchical structure is asymmetric and is a major drawback.	The network structure is more symmetric than hierarchical structure.	The relational structure is more symmetric than network and hierarchical structure.
7.	Information is replicated in hierarchical model which leads to inconsistency and wastage of storage space.	It offers more data consistency than hierarchical model.	It offers more data consistency than other two models.

2.4 LEVELS OF ABSTRACTION (S-17)

- The main purpose of a database is to provide users with an abstract view of the data.
- Data abstraction refers to, "the hiding of certain details of how the data are stored and maintained".

- There are three levels of data abstraction.
  1. View (External) Level
  2. Conceptual (Logical) Level
  3. Physical (Internal) Level

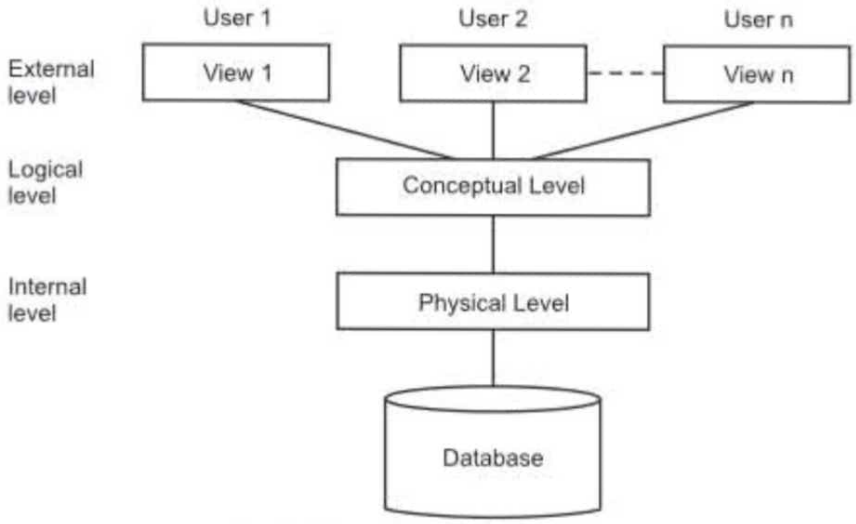


Fig. 2.6: Levels of data abstraction

1. View Level:

- View level is the highest level of data abstraction. Each view in the view level is used to describe a part of the database that a particular user group is interested in and hides the rest of the database from that user group.
- For example, there is an Employee table which has emp_id, name, designation, salary. Suppose user 'A' want to view only emp_id and name and user 'B' want to view only emp_id, designation, salary. We can provide two different views to user 'A' and user 'B' to satisfy their requirements.

2. Conceptual level:

- It is the next higher level of data abstruction. It is used by the DBA to describe what data are stored in the database and the relationship between those data. The structure of a table is specified in terms of attributes. Attribute type is specified in terms of logical structure of the database. It also describes the relationship between the different tables stored in the database.

3. Physical Level:

- It is the lowest level of data abstraction. It describes in detail how the data is actually stored in the database using complex low level data structure. It also describes the access paths for the database. The database system hides many of these details from the database programmers and the end users.

2.5 DATABASE SCHEMA AND INSTANCE (S-17; W-17)

- Schema is a structural representation of a database. Instance is the database is the data stored in the database at a particular moment of time.
- Schema does not frequently change. Instance is a dynamic concept. Hence it changes frequently.

Database Schema:

Column	Emp_Id	Emp_Name	Designation	Address	Salary
Data Type	Numeric	Text	Text	Text	Numeric

Database Instance:

Emp_Id	Emp_Name	Designation	Address	Salary
11	Avinash	Manager	Main Road	150000
12	Ramchandra	Programmer	East Road	100000
13	Sameer	Programmer	Main Road	100000
14	Kalpana	Programmer	West Road	100000
15	Joseph	Programmer	Main Road	100000

2.5 DATA INDEPENDENCE

- Data Independence is defined as the characteristic of a database system to change the schema at one level without having to change the schema at the next higher level.
- There are two levels of data independence: Physical Data Independence and Logical Data Independence.

2.5.1 Physical Data Independence

- It is the ability to change the internal schema without having to change the conceptual or external schema. Physical data independence is easy to achieve as compared to logical data independence. If we want to upgrade or change the physical storage device, then the change done would be incorporated by the mapping between the conceptual and internal levels.

2.5.2 Logical Data Independence

- It is the ability to change the logical schema of a database without having to change the view schema of the database. Application programs at the external level do not get affected by changes made at the logical level. For example, add/modify/delete a new attribute, entity or relationship is possible without any change in the existing application programs.

2.5.3 Advantages of Data Independence

1. Improvement in database security.
2. The cost of maintaining database system is reduced.
3. It allows the users to focus on general structure without worrying about its internal implementation.
4. Alterations in data structure does not need to alter the application programs.

Difference between Physical Data Independence and Logical Data Independence:

Sr. No.	Physical Data Independence	Logical Data Independence
1.	It is related with immunity of conceptual schema to changes in the internal schema.	It is related with immunity of external schemas to changes in conceptual schema.
2.	This does not require change to conceptual or external schemas.	This does not require changes to external schema or rewrites of application programs.
3.	In this internal schema changes.	In this conceptual schema changes.

2.6 STRUCTURE OF DBMS

- A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The components of a database system can be broadly divided into Storage Manager and Query Processor.
- Fig. 2.7 shows Overall structure of DBMS.

1. **Query Processor Components:** It has following sub-components.
   (i) **DML compiler and organizer:** When a DML statement is given, it translates the DML statement to low level instructions which a query evaluation engine understands. It also transforms the user's request into more efficient form for the query evaluation engine.
   (ii) **Compiler and Linker:** It converts the DML statements embedded in application program to normal procedure call in host language, so that it interacts with DML compiler.
   (iii) **DDL Interpreter:** It translates DDL statements into a set of tables and records them in the form of metadata or data dictionary.
   (iv) **Query Evaluation Engine:** It executes the low-level instructions generated by the DML compiler.
2. **Storage Manager Components:** They provide an interface between the low-level data stored in database and application programs and the queries submitted to the system.

Following are the sub-components:

(i) **Authorization and Integrity Manager:** It first checks the authority of users to access data. Then it checks whether the integrity constraints are satisfied or not.

(ii) **Transaction Manager:** It ensures that the database remains in a consistent state inspite of system failure. It also ensures that concurrent transaction execution proceeds without conflicting.

(iii) **File Manager:** It manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

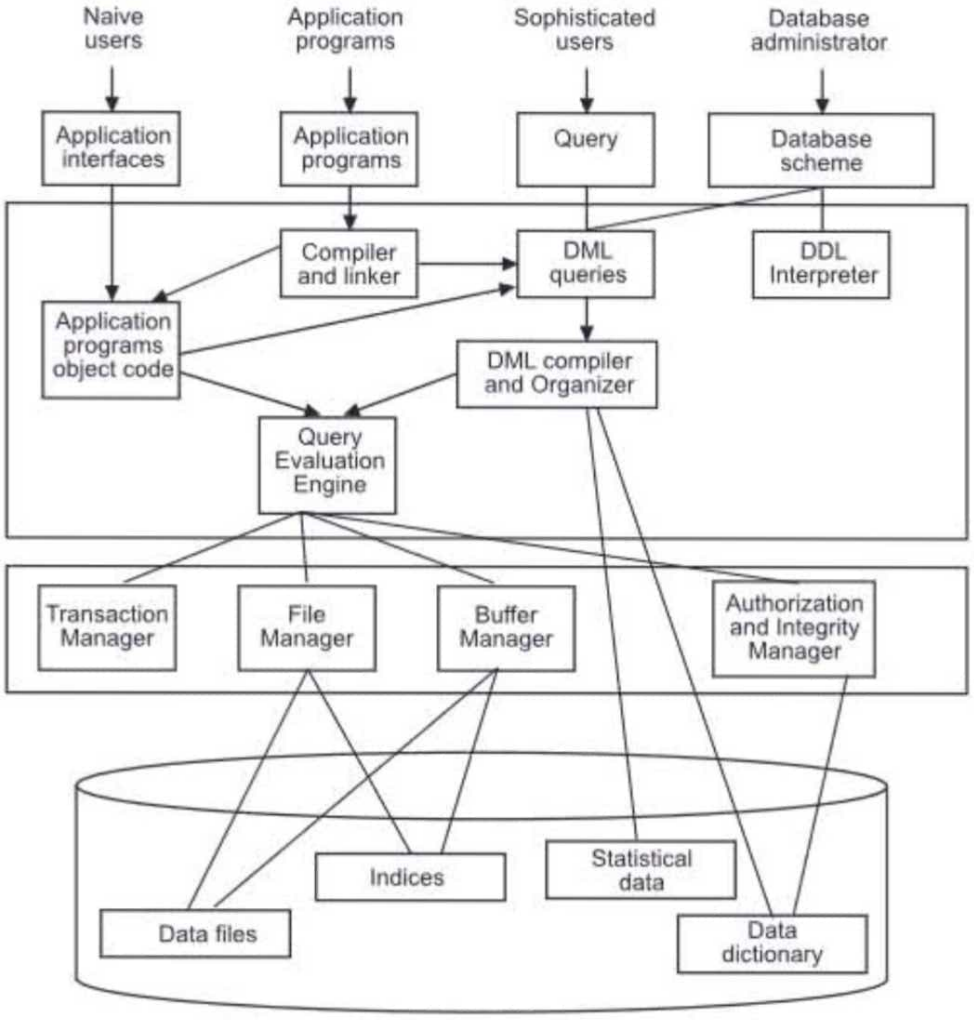


Fig. 2.7: Overall System Structure

(iv) **Buffer Manager:** It is responsible for fetching data from disk storage into main memory and deciding which data to cache in memory. Other than these two main parts, there are some data structures available which resides in DBMS.

1. **Data files:** It stores the database.
2. **Data Dictionary:** It stores metadata (data about data) about the structure of the database.
3. **Indices:** This provides fast access to the data items that hold particular values.
4. **Statistical data:** It stores statistical information about the data in the database.

2.7 USERS OF DBMS

- The users of Database System can be classified on the basis of degree of expertise or the mode of their interactions with DBMS.
- There are four groups of users of DBMS:
  1. Database Designers
  2. Application Programmers
  3. Sophisticated Users
  4. End Users

2.7.1 Database Designers

- There are two main database designers.
  (i) Database Manager
  (ii) Database Administrator

2.7.1.1 Database Manager

- A Database Manager is supervising authority over the database administrator. A Database Manager manages and updates a database. The Database Manager has some knowledge about the database, but need not be having the same expertise as the database administrator. Actually the database administrator takes care of the database. The database administrator can create, delete and update a database.

2.7.1.2 Database Administrator (DBA) (S-18)

- The DBA is an individual person or group of persons (a team) having a good knowledge in data processing. DBA must have sufficient technical knowledge to make decisions in the organization.
- The DBA must manage the staff to ensure orderly development of the database project, to satisfy database users and to plan for future database requirements.

Responsibilities of DBA:

1. DBA decides conceptual and internal schema. The data dictionary is created by DBA to help users to query the dictionary.
2. DBA defines authorization checks and validation procedures. DBA checks validity before giving access.
3. DBA decides the storage structures and access strategy, i.e. which user can retrieve what.
4. DBA defines security and integrity rules.
5. DBA monitors the system performance.
6. DBA defines strategies for backups and recovery.

2.7.2 Application Programmers

- They are computer professionals who are responsible for the development of application programs.
- They use a general purpose programming language for the development of application programs.
- By using these programs, these people handle or manipulate the databases.

2.7.3 Sophisticated Users

- These people are also computer professionals but they do not write programs. To interact with the system, they use query languages like SQL.

2.7.4 Specialized Users

- They are sophisticated users who write specialized database application programs.
- For example, Computer Aided Design system, Knowledge base and Expert Systems etc.

2.7.5 End users

- End users are unaware of the presence of database system or any other system. These users are called as unsophisticated users who interact with the system through already written permanent programs.
- They are also known as naive users.

2.8 ADVANTAGES AND DISADVATNAGES OF DBMS (S-17, 18)

Advantages of DBMS:

1. In DBMS, the data can be shared among all authorized users.
2. The centralized control of DBMS about adequate checks can be incorporated to provide data integrity. Data integrity means the data is both accurate and consistent.

3. Inconsistency of data is avoided.
4. In DBMS, the database system is aware of the redundancy and it assumes the responsibility for propagating updates.
5. The DBMS uses different techniques to access the stored data very efficiently.
6. Program and data interdependency is decreased.
7. Security to data is provided, which ensures that proper access procedures are followed. It includes authentication schemes for access to the database and additional checks before- permitting access to sensitive data.
8. The DBMS has backup and recovery subsystem which provides facilities for recovering from hardware or software failures.
9. Data quality is maintained.
10. Maintenance efforts are reduced to minimal in database systems due to independence of data and application programs.
11. User can monitor database performance.
12. Cost of software development is reduced.
13. The overall productivity of the programmer is increased.

Disadvantages of DBMS:

1. **Cost of investment in hardware and software :** CPU having high speed of data processing and large memory size is needed.
2. **Difficulty of data migration :** It is difficult to convert data files into database.
3. **Requirement of technical staff :** Trained technical persons are required to handle the operations on DBMS.
4. **Impact of a failure :** The data is valuable and it is integrated into a single database. Due to some failure if a database is corrupted, then valuable data may be lost.
5. **Cost of staff training :** Training of staff to manage the DBMS operations requires some amount.

Summary

- A DBMS (Database Management System) is a complex software system that is used to manage, store and manipulate data.
- Data can be defined as a representation of facts, concepts or instructions.
- There are limitations in file processing system such as: separated and isolated data, data redundancy, security problem etc.
- A model is an abstraction process that hides the details which are not required while highlighting details required to the applications at hand.
- Relational Model shows collection of tables to represent both data and the relationship.

- In Network Model, data is represented by collection of records and the relationship among the data is represented by links.
- In Hierarchical Model, data is organized into a tree-like structure in which each child node can have only one parent node.
- Internal level describes how the data is actually stored.
- Logical level describes what data are stored in the database.
- View level is used to describe a part of the database that a particular user group is interested in and hides the remaining part of the database from that user group.
- Data independence means that the application is independent of the storage structure and access strategy of data.
- The functional components of a database system can be broadly divided into the storage manager and the query processor components.
- Query processor components: DML compiler and Organizer, Compiler and Linker, DDL Interpreter, Query Evaluation Engine.
- Storage manager components: Authorization and Integrity Manager, Transaction Manager, File Manager, Buffer Manager.
- There are four groups of users of DBMS: database designers, application programmers, sophisticated users, end users.
- Some of the advantages of DBMS:
  (i) Inconsistency of data is avoided.
  (ii) Program and data interdependency is decreased.
  (iii) Security to data is provided.
  (iv) Data quality is maintained.

Check Your Understanding

Q.1. Multiple Choice Questions:

1. Data is better organized and is easily accessible by using .....
   (a) DBMS (b) FMS
   (c) EMS (d) all
2. Which of the following is an advantage of DBMS?
   (a) reduction in redundancy (b) avoidance of inconsistency
   (c) security enforcement (d) all above
3. In ..... model, we have a strict parent-child relationship only.
   (a) hierarchical (b) network
   (c) relational (d) all

4. ..... is a collection of different fields that provide information on an entity.
   (a) field (b) record
   (c) data item (d) none of these
5. The data in the databse are both shared and .....
   (a) isolated (b) attached
   (c) integrated (d) none of these
6. ..... are used in describing data at the logical and view levels.
   (a) Record based logical models (b) Object based logical models
   (c) Physical data models (d) none of these
7. ..... level is used to describe what data are stored in the database.
   (a) Conceptual (b) Physical
   (c) View (d) none of these
8. ..... executes the low-level instructions generated by the DML compiler.
   (a) DDL interprefer (b) Query evaluation engine
   (c) Storage manager (d) none of these
9. The database ..... can create, delete and update a database.
   (a) manager (b) administrator
   (c) operator (d) none of these
10. Which of the following is not an abstraction level ?
    (a) view (b) conceptual
    (c) physical (d) complex

Ans.: (1) - a (2) - d (3) - a (4) - b
(5) - c (6) - a (7) - a (8) - b
(9) – b (10) - d

Q. 2. State True/False.

1. Processed data is called as information.
2. Data Redundancy is one of the limitations of the file processing system.
3. DBMS needs very little preliminary design.
4. Object based data models has E-R model as its subtype.
5. In relational model, model data is organized into a tree-like structure.
6. Logical level deals with the way a particular user views the data from the database.
7. Logical data independence is the ability to change the internal schema of a database without having to change the logical schema.
8. A database manager is supervising authority over the database administrator.

9. File manager is responsible for fetching data from disk storage into main memory and deciding what data to cache in memory.
10. In DBMS, the data can be shared by all authorized users.

Ans.: (1) True (2) True (3) False (4) True
(5) False (6) False (7) False (8) True
(9) False (10) True

Practice Questions

Q.1. Answer the following questions in brief.

1. State any two differences between data and information.
2. State any two limitations of file processing system.
3. What is a data model ?
4. State any one difference between hierarchical and network model.
5. What is physical data independence ?
6. What is logical data independence ?
7. What are application programmers ?
8. What is the role of database manager ?
9. State any two advantages of DBMS.

Q.2. Answer the following questions.

1. With suitable diagram describe levels of abstraction.
2. Explain the overall structure of DBMS.
3. Describe drawbacks of file processing system.
4. Differentiate between file system and database system.
5. What is DBA ?
6. Describe data independence in detail.
7. Compare between hierarchical and network model.
8. Compare between network and relational model.
9. What are the advantages of DBMS.

Q.3. Define the terms.

(a) Database (b) Data
(c) Information (d) DBA
(e) Database Manager (f) Data redundancy
(g) Data Model (h) Physical data model
(i) Record based logical model (j) Relational model
(k) data Abstraction

Previous Exams Questions

Summer 2017

1. The capacity to change the physical schema without having change to logical or view schema is known as ..... . [1 M]
   (i) physical data independence
   (ii) logical data independence
   (iii) data independence
   (iv) None of the above

Ans.: Refer to Section 2.5.

2. Write short note on data abstraction and data independence. [5 M]

Ans.: Refer to Sections 2.4 and 2.5.

3. What is data model ? Write short note on network data model. [4 M]

Ans.: Refer to Section 2.3.1.

4. Write a note on 'Uses of DBMS'. [4 M]

Ans.: Refer to Section 2.8.

Winter 2017

1. ..... is the capacity to change the schema at one level of database system without having to change schema at next higher level. [1 M]
   (i) Logical data independence (ii) Physical data independence
   (iii) Data independence (iv) None of the above

Ans.: Refer to Section 2.5.

2. State the user of DBMS. [1 M]

Ans.: Refer to Section 2.7.

3. Differentiate between File system and Database management system. [5 M]

Ans.: Refer to Section 2.2.

4. What is data model ? Write short note on network relational data model. [4 M]

Ans.: Refer to Section 2.3.2.

Summer 2018

1. A set of logically related record forms a ..... . [1 M]
   (i) database (ii) file
   (iii) record (iv) None

Ans.: Refer to Section 2.1.

2. Define : instance of database. [1 M]

Ans.: Refer to Section 2.4.

3. What is DBA ? Explain any two function of DBA. [3 M]

Ans.: Refer to Section 2.7.1.2.

4. What are advantages and disadvantages of DBMS ? [3 M]

Ans.: Refer to Section 2.18.

❖❖❖

3...

Conceptual Design (E-R Model)

Objectives...

- To understand the basic concepts in database design.
- To know about E-R data model.
- To understand the basic concepts in E-R model and its use in database design.
- To have knowledge about different notations used in E-R model.

3.1 OVERVIEW OF DB (DATABASE) DESIGN

- A good database design begins with identifying a list of the data that you want to make part of your database. It also focus on, what you want to do with the database later on. The task of designing a database is complex. There is a big role of the needs of the users in this process. The main objectives of database designing are to produce logical and physical design model for the new proposed system.
- The logical model has a main focus on data requirements and the data to be stored independent of physical considerations. The physical design model involves translating logical design onto physical media.
- Properly designed database improves the performance, is easy to maintain, is cost effective and improves the data consistency.
- Fig. 3.1 illustrates the phases of database design. There are three phases: Requirements Collection and Analysis, Designing of database (Logical Model, Physical Model), Implementation.
  1. **Requirements collection and analysis:** The database designer interacts with potential users to understand their data requirements in this phase. The result of this phase is user requirements specification.
  2. **Designing of database:**
     Logical design: In this stage, the designer maps the high-level conceptual schema onto the implementation data model of the DBMS.

Physical design: In this stage, the physical features of the database are specified.

3. **Implementation:** Data conversation and loading is concerned with importing and converting data from the old system into the new database. The main purpose of testing is identification of errors in the new system. It also checks whether the database meets the requirement specifications.

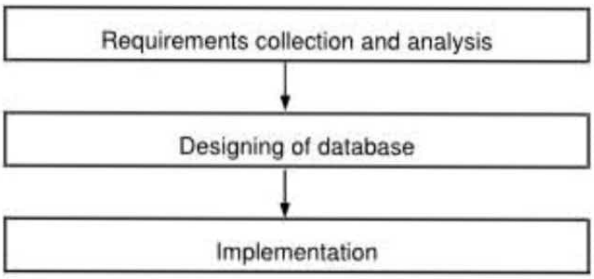


Fig. 3.1: Phases of Database Design

3.2 E-R DATA MODEL

- The E-R model is an abstract data model, which defines a data or information structure which can be implemented in a database. Entity–relationship modelling was developed by Peter Chen and published in a 1976 proper. E-R model is based on the concept of real-world entities and the relationship between them. It analyzes data requirements systematically and creates a well-designed database. Hence, it is recommended to complete E-R modelling before implementing the database.

3.2.1 Entities (S-17, W-17, S-18)

- An entity is a thing or object that is distinguishable from other. For example, in a company database, employees, departments can be considered as entities. An entity can be an object with a physical existence (e.g. student, book etc) or it can be an object with a conceptual existence (e.g. subject, project etc). An entity can be either living or non-living object. Each entity has some properties that describe it.
- Entities are represented by rectangles.

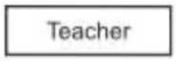
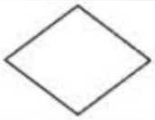
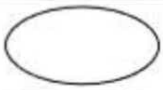
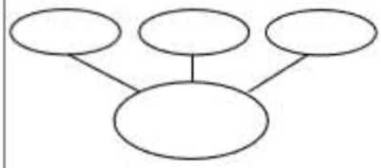
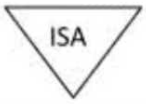


Fig. 3.2: Teacher

- **Entity Set:** An entity set is a set of all entities of the same type. In an entity set, entities share the same set of properties.
- For example, the set of all books is defined as entity set Book. Here all books in an entity set Book share properties such as isbn number, title, author, year, edition, publisher, price.

3.2.1.1 E-R Diagram Notations (Chen Notation)

Notation	Meaning
	Entity
	Relationship
	Attribute
	Derived Attribute
	Multivalued Attribute
	Composite Attribute
	Weak Entity
	Weak Relationships
	Specialization(Top Down) Generalization (Bottom Up)
	Total participation of entity
	Key attribute

Types of Entity Sets:

- There are two types of entity set:
  1. **Weak Entity set:** An entity set that does not have sufficient attributes of its own to form a primary key is known as weak entity set. A primary key (an attribute or combination of attributes or properties) is used to uniquely identify an entity from an entity set. Weak entity is represented as double rectangle.

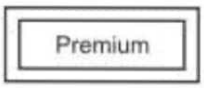


Fig. 3.3: Premium entity

The weak entity is also known as dependent entity because its existence is dependent on another entity.

  2. **Strong Entity set:** An entity set that has sufficient attributes (properties) to form a primary key is known as Strong Entity Set. Strong entity is represented by a rectangle.

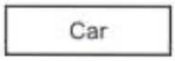


Fig. 3.4: Car Entity

Comparison between Strong entity set and Weak entity set:

- A member of strong entity set is called dominant entity, whereas a member of weak entity set is called as subordinate entity.
- Strong entity set contains a primary key (represented by an underline).
  Weak entity set contains a discriminator (represented by a dashed underline).
- The relationship between two strong entity sets is represented by a diamond symbol.
- The relationship between a strong and a weak entity set is shown by a double diamond symbol. It is known as the identifying relationship.
- Total participation in the relationship may or may not exist in case of strong entity set. In case of weak entity set total participation in the identifying relationship always exists.

Example:

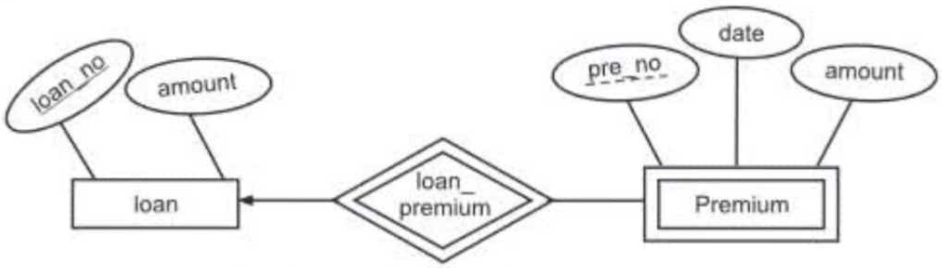


Fig. 3.5: E-R diagram with weak entity set

3.2.2 Attributes (W-17)

- An entity can be described with the help of some properties, that are known as attributes of an entity.
- For example, a student entity can have attributes such as roll number, name, class, division, data of birth, address etc.
- **Domain:** It is a set of all possible values that an attribute can have. For example, the domain of attribute roll number is the set of all positive numbers. Attributes are represented by an ellipse.

Types of attributes:

1. **Simple (atomic) attribute:** The attribute which cannot be splitted into meaningful subparts is known as simple or atomic attribute.
   For example, roll_ number is an atomic or simple attribute.
2. **Composite attribute:** The attribute which can be splitted into meaningful subparts in known as composite attribute.
   For example, emp_name attribute could be structured as composite attribute consisting of first_ name, middle_name and surname.
   Composite attributes are represented by ellipses that are linked (connected) with an ellipse.

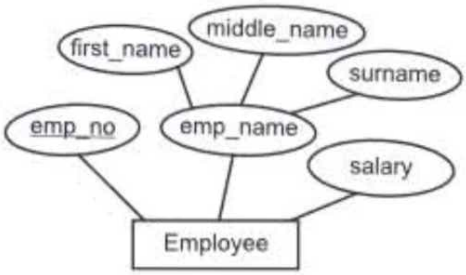


Fig. 3.6: Employee entity having emp_name as composite attribute

3. **Single-valued attribute:** The attribute that has a single value for a particular entity is called as a single valued attribute.
   For example, roll_number of a student entity is a single-valued attribute.
4. **Multi-valued attribute:** An attribute that holds multiple values for a single entity is known as multi-valued attributed. Multi-valued attributes are represented by double ellipse.
   For example, phone number of an employee entity set can be multi-valued attribute.
5. **Stored attribute:** The attribute which cannot be derived from other attributes is called as stored attribute.
   For example, date_of_birth is a stored attribute.

6. **Derived attribute:** The attribute whose value can be derived from the values of other related attributes is known as a derived attribute.
   For example, age of an employee entity is a derived attribute. It is represented by dashed ellipse.
7. **Null attribute:** An attribute does not have a value for an entity is called as null attribute. Null value is used to represent the null attribute. Null value means missing or unknown.

3.2.3 Relationship sets

- A relationship is an association among two or more entities. For example, Chetan works in Hindi department.
- A relationship set is a set of relationships of the same type.
- A relationship is represented by diamond shape in E-R diagram.
- **Degree of a relationship type:** The degree of a relationship type is the number of participating entities in a relationship.
- There are three types of degrees of relationships, namely unary, binary and ternary.

(i) A relationship that involves one entity type is called unary relationship. A unary relationship is also called recursive in which a relationship exists between occurrences of the same entity set.

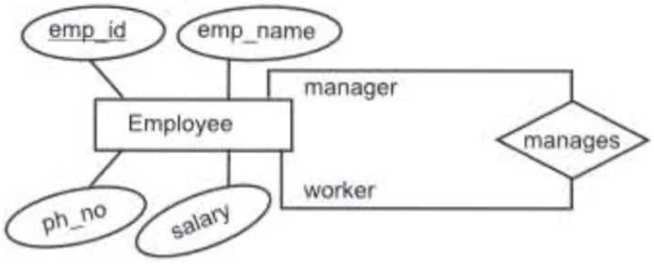


Fig. 3.7: Unary relationship

- Fig. 3.7 shows the role indicators manager and worker between the Employee-entity set and the manages relationship set.

(ii) A relationship between two entity types is known as binary relationship. Binary relationship is the most common relationship type.

- Fig. 3.8 shows binary relationship.

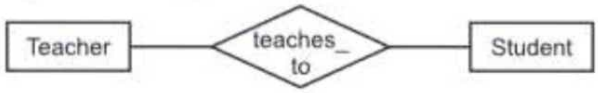


Fig. 3.8: Binary relationship

(iii) A ternary relationship is formed when three entities participate in the relationship.

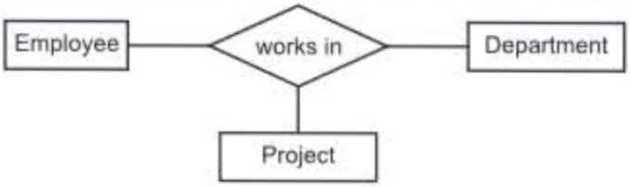


Fig. 3.9: Ternary Relationship

3.2.3.1 Mapping Cardinalities

- Mapping cardinalities describe the number of entities to which another entity can be, linked via a relationship set. Mapping cardinalities must be one of the following:
  1. **One-to-One:** An entity in A is related to at most one entity in B, and an entity in B is related to at most one entity in A.

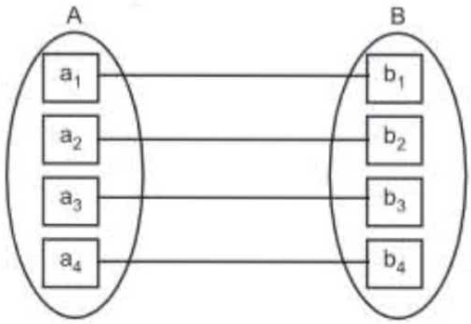


Fig. 3.10: One-to-One Mapping

  2. **One-to-Many:** An entity in A is related to any number of entities in B, but an entity in B is related to at most one entity in A.

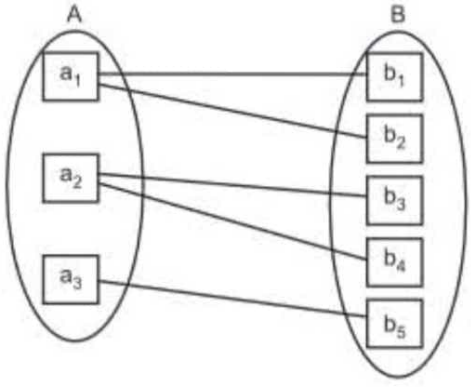


Fig. 3.11: One to-Many Mapping

  3. **Many-to-One:** An entity in A is related to at most one entity in B, but an entity in B is related to any number of entities in A.

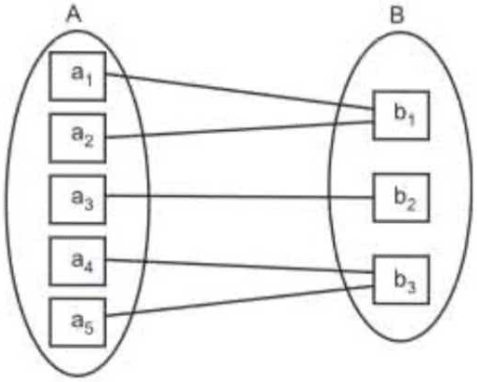


Fig. 3.12: Many-to-One Mapping

4. **Many-to-Many:** An entity in A is related to any number of entities in B and an entity in B is related to any number of entities is A.

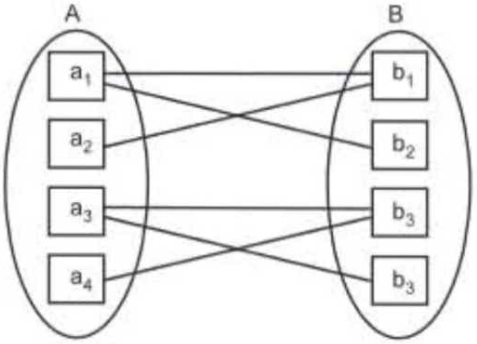


Fig. 3.13: Many-to-Many Mapping

- To find appropriate mapping cardinality – for a relationship set depends on the actual problem (situation) that the relationship set is dealing with.
  For example, consider the *treated–by* relationship set. If in a particular situation, a patient can be treated by one doctor, and a doctor can treat several patients, then the relationship set from doctor to patient is one-to-many. If a patient can be treated by several doctors, then the relationship set is many-to-many.
- E-R diagram can be drawn by using notations like Chen Notation, Crow's foot notation. The Chen E-R diagram notation is considered to present a detailed way of representing entities and relationships.
- The Crow's foot notation was invented by Gordon Everest. Initially it was called the "inverted arrow". Now it is known as a "fork". A fork or Crow's foot represents "many" by its many "toes".

(i) The Chen E-R diagram notation

(a) One-to-One relationship

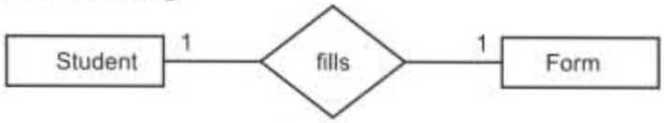


Fig. 3.14 (a)

(b) One-to-Many relationship

Fig. 3.14 (b)

(c) Many-to-One relationship

Fig. 3.14 (c)

(d) Many-to-Many relationship

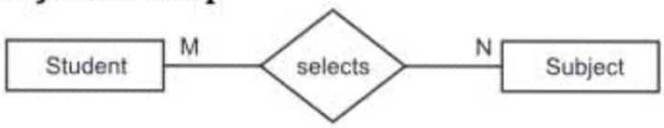


Fig. 3.14 (d)

(e) Total participation

Fig. 3.14 (e)

- In the Chen Notation, total participation is represented by a double line. It means that there can be teachers who don't teach any students, but there is no student that not taught by a teacher.

(ii) The Crow's foot notation (see Fig. 3.15):

(a) Entity

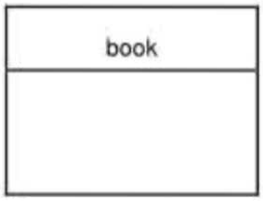


Fig. 3.15 (a)

(b) Attributes

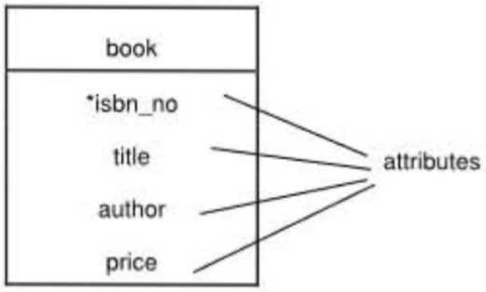


Fig. 3.15 (b)

(c) Relationship

Fig. 3.15 (c)

- It is labelled with a word/words.

Cardinality: It is referred as mandatory part of relationship. It refers to the maximum number of times an instance of one entity set can be linked with instances in the related entity set.

Modality: It is an optional part of relationship. It refers to the least number of times an instance of one entity set can be related with instances in the related entity set.

These two (mandatory and optional) parts appear in a specific order.

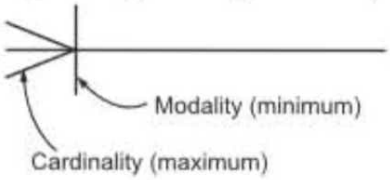


Fig. 3.15 (d)

There are four possible edges to the relationship.

(i) Zero or many

Fig. 3.15 (e)

(ii) One or many

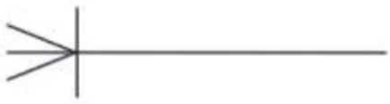


Fig. 3.15 (f)

(iii) One and only one

Fig. 3.15 (g)

(iv) Zero or one

Fig. 3.15 (h)

(d) One-to-One relationship

Fig. 3.15 (i)

(e) One-to-Many relationship

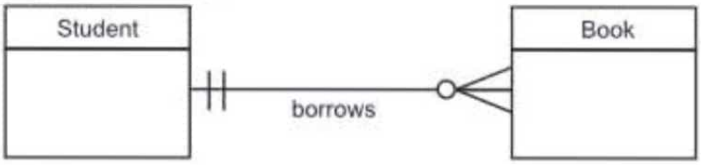


Fig. 3.15 (j)

(f) Many-to-One relationship

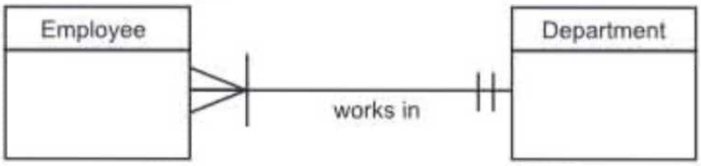


Fig. 3.15 (k)

(g) Many-to-Many relationship

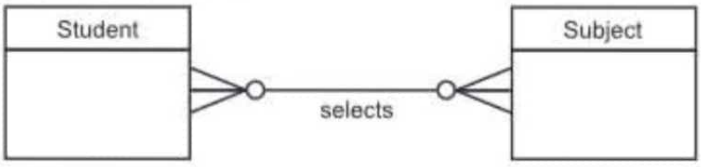


Fig. 3.15 (l)

3.3 ADDITIONAL CONSTRAINTS (S-17, S-18)

3.3.1 Key Constraints

- A key is a set of attributes that uniquely identifies an entity within an entity set. Keys also help us to uniquely identify relationships and thus distinguish the relationships from each other.
- In case of the multi-attribute keys, it is possible for two entities to agree on some, but not all, of the key attributes. There must be a key for every entity set.
- **Superkey:** A superkey is a set of one or more attributes (taken collectively) allows to identify a unique entity within an entity set.
  For example, the emp_id attribute of the employee entity set is a super key. Also, the combination of emp_id and emp_name is a superkey for the employee entity set. The emp_name attribute is not a super key, because many persons may have the same name.
- **Candidate key:** The minimal superkey is known as candidate key. {emp-id} and {emp-id, emp-name} are candidate keys.
- **Primary key:** It is the candidate key chosen by the database designer.

3.3.2 Participation Constraints

- There are two types of Participation Constraints.
  1. **Total participation:** Every entity in an entity set E is involved in the relationship. It is represented by double lines.
  2. **Partial participation:** Not all entities in an entity set E are involved in the relationship. It is represented by single lines. In Fig. 3.16 loan and customer are two entities sets and relationship set is *borrows*.
     Every customer may or may not take the loan, so customer entity set is partially participated. Each loan entity from the Loan entity set is associated with a customer entity from the Customer entity set.

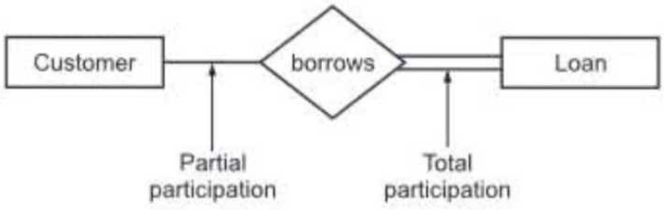


Fig. 3.16: Total participation

3.3.3 Extended Features (S-17, W-17)

- **Specialization:** It is the process of identifying subsets of an entity (superclass) that share some distinguishing characteristic.

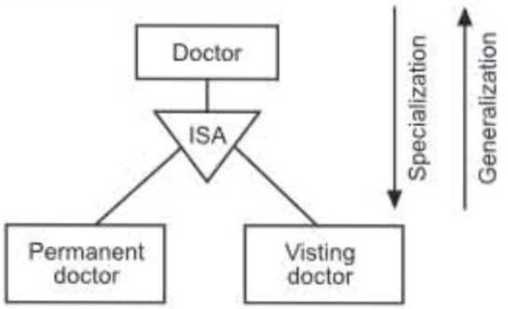


Fig. 3.17: Specilization

- In Fig. 3.17, there are two subgroups that exist within a doctor (superclass) entity set. i.e. permanent doctor and visiting doctor.
- The two subgroups formed are distinct. Permanent doctor earns salary and visiting doctor is paid with consultation fee.
- **Generalization:** It is the process of identifying common characteristics among the sub-classes (lower-level entity set) to synthesize into a single, higher-level entity set. It works in bottom-up manner while specialization works in top-down-manner.
- In the Fig. 3.17, the generalization line implies that a doctor has to be either a permanent doctor or a visiting doctor.
- **Aggregation:** It is an abstraction through which relationship is treated as higher level entity set.

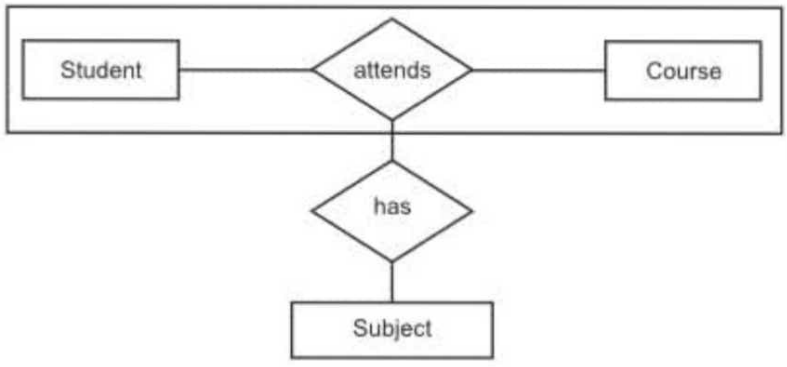


Fig. 3.18: Aggregation

- In the fig. 3.18, attends is treated as higher level entity set. Then a binary relationship between attends and subject is formed.

3.3.4 Case Studies (S-17, W-17)

1. E-R diagram for flat booking system.

   DreamHome is an agency for flat booking and it has number of builders and agents who are jointly working. A customer can buy a flat for residential or commercial purpose. An agent is given commission if customers are approached to the agency through agent. Agent provides information regarding various flats within the location.

   Draw an E-R diagram for the above case .

Solution:

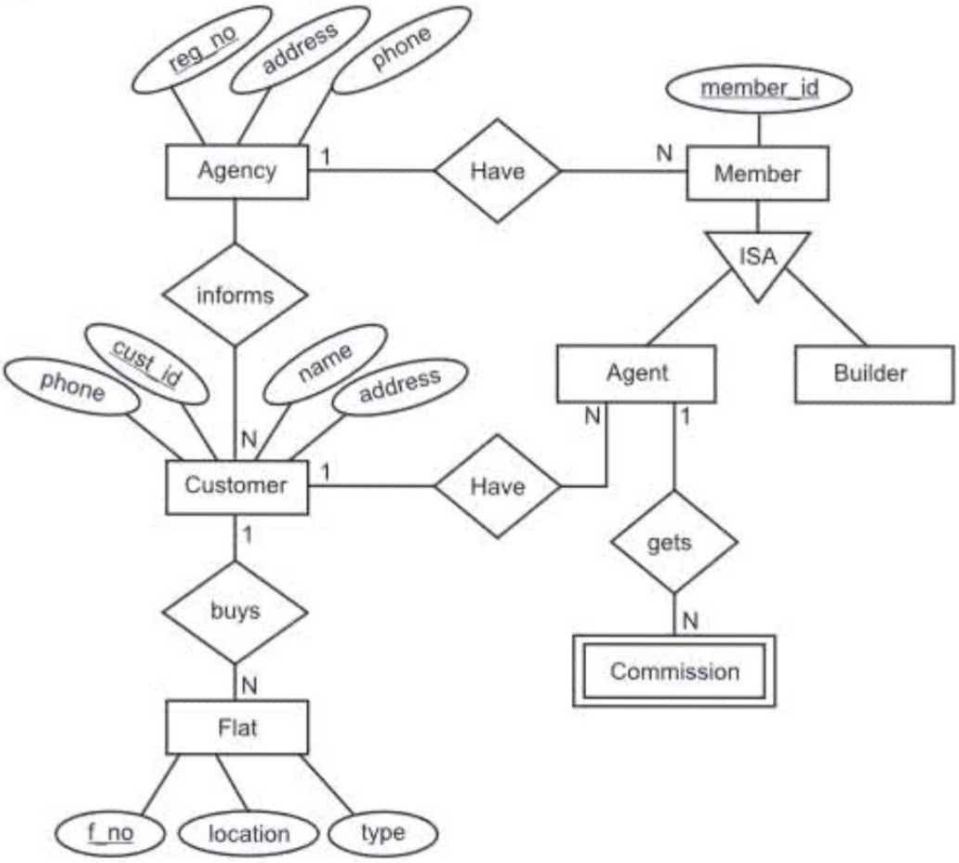


Fig. 3.19

2. A car insurance company has a set of customers. Each customer owns one or more cars. Each car can have zero to any number of recorded accidents.

   Draw an E-R diagram for the above case.

Solution:

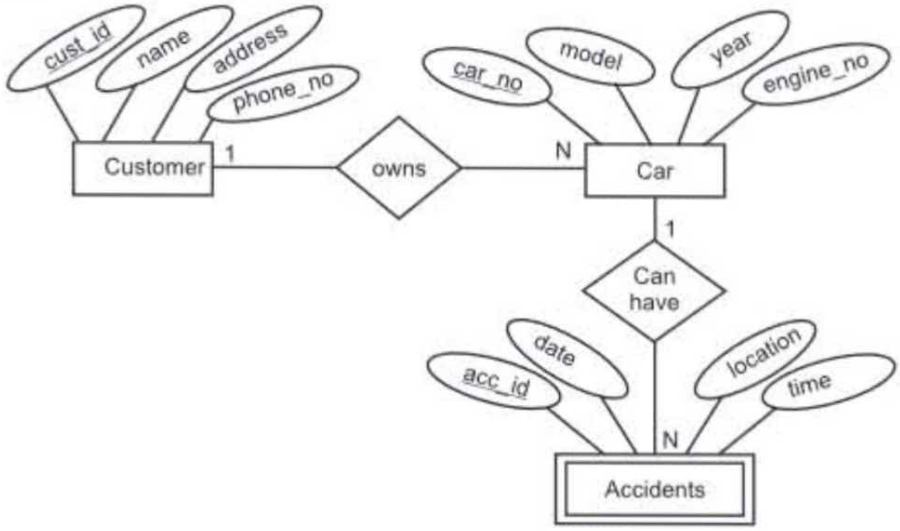


Fig. 3.20

3. In a nursery, the customer purchase plants.
   The plants are type: flowering, non-flowering. Plants are provided nutrients with some quantity. Nutrients are pesticides, watering, manure.
   Draw on E-R diagram for the case.

Solution:

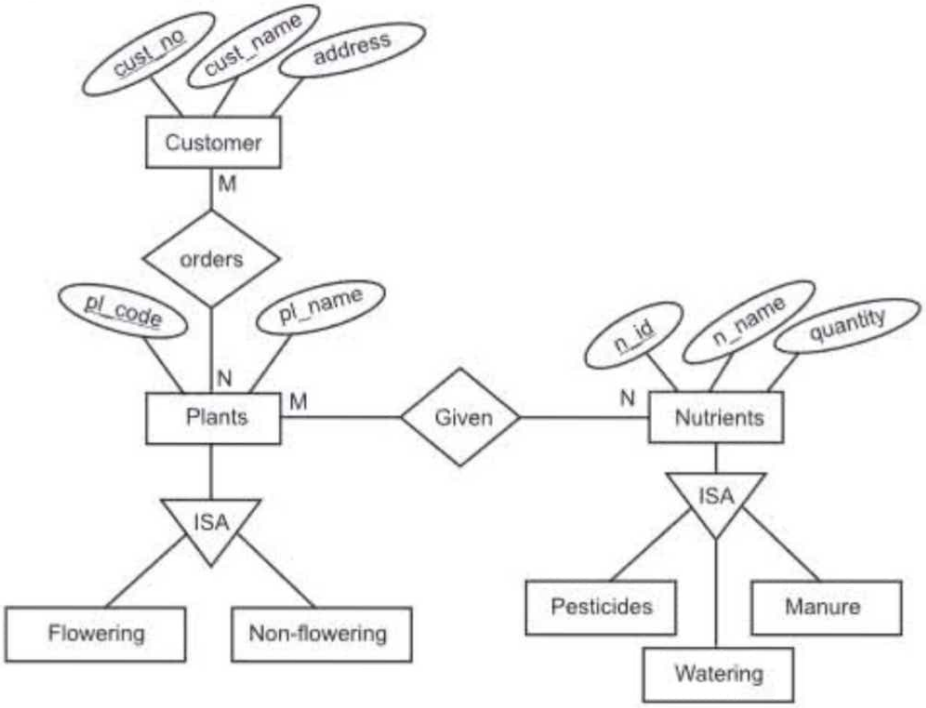


Fig. 3.21

4. In an Airline Reservation System, passenger can book one or more tickets. Flight are associated with tickets.

   Draw an E-R diagram.

   Solution:

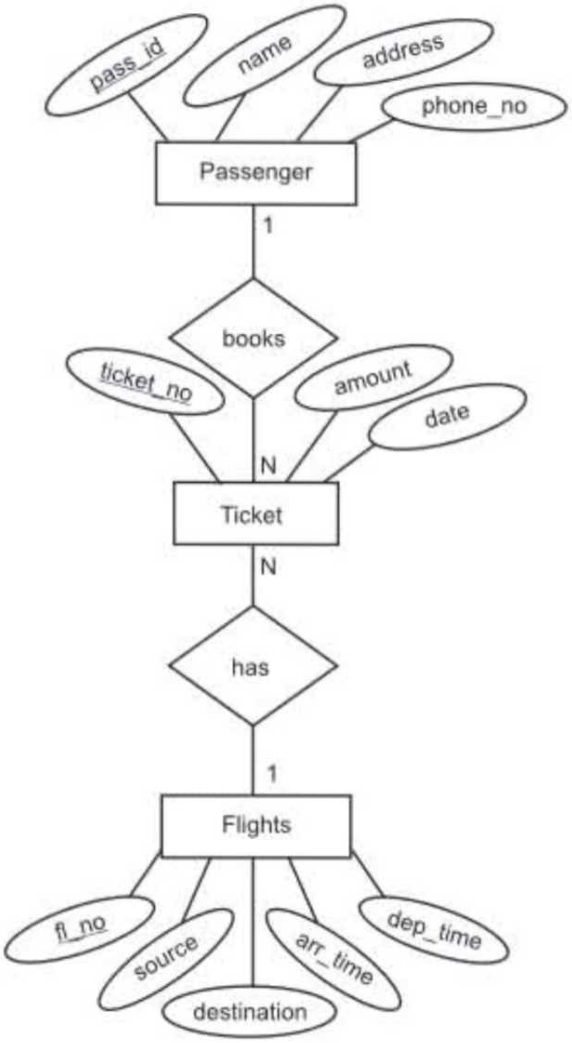


Fig. 3.22

5. In Online Hotel Reservation System, Customer makes reservation and pays the payment. Through the reservation, customer books the room. Draw an E-R diagram.

Solution:

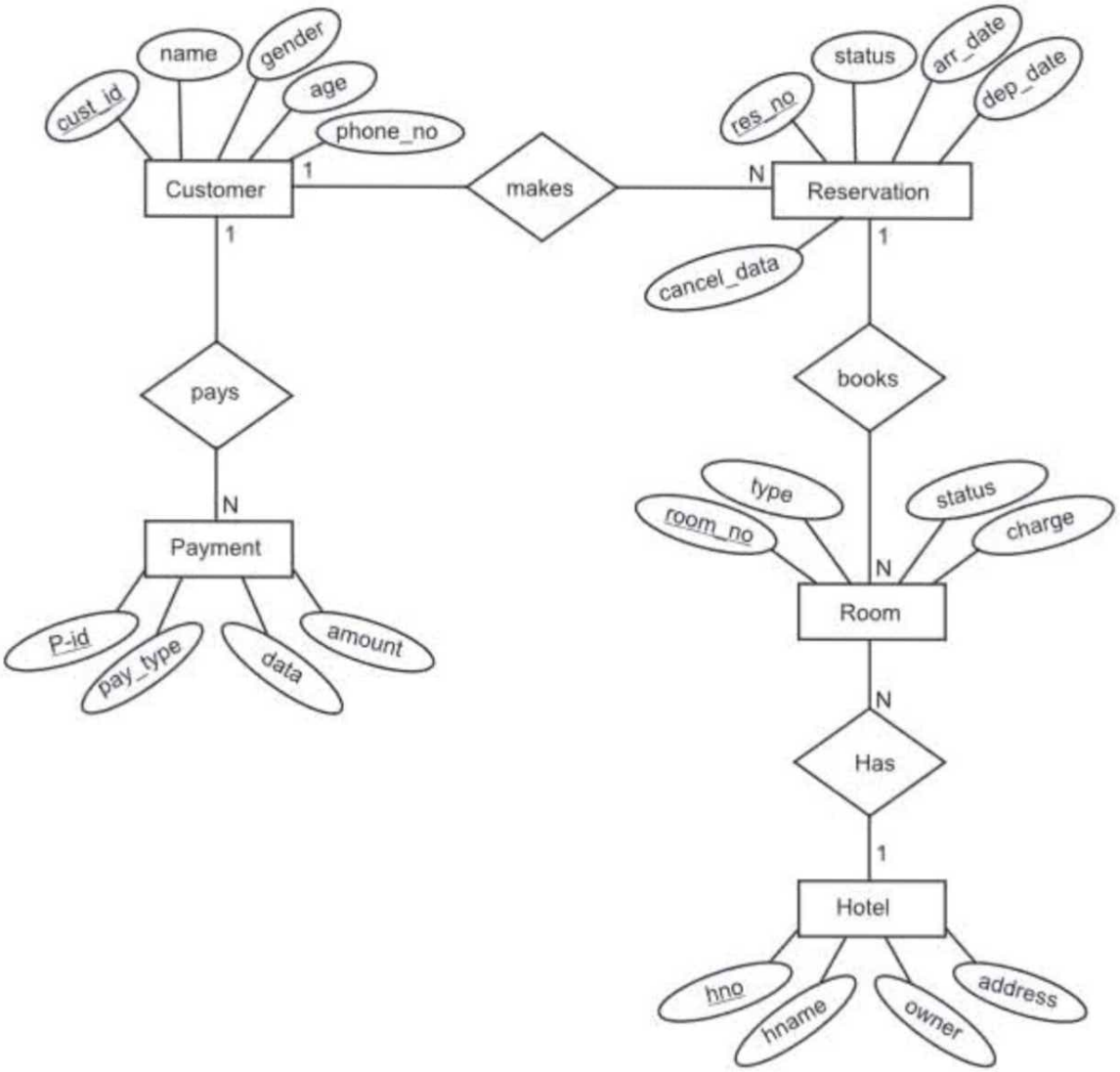


Fig. 3.23

6. In an On-line bookshopping application, user can order products, which are then placed in a cart. After confirmation a bill is prepared. Each bill has a payment details. Draw an E-R diagram.

Solution:

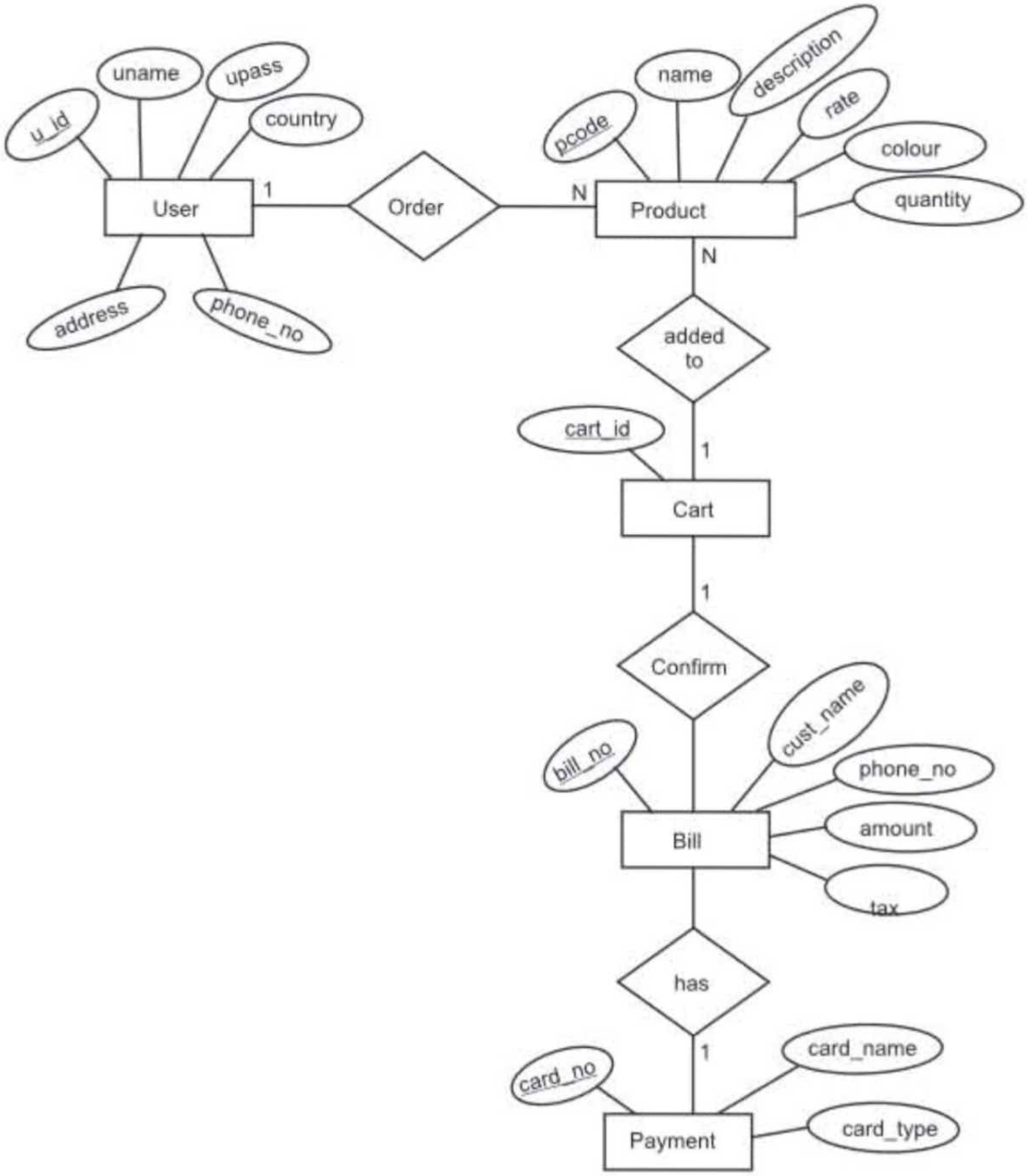


Fig. 3.24

7. In a multiplex theatre there are many screens. Each screen runs many shows. A movie can have one or more shows. Each show has many tickets. A customer can buy one or more tickets. Draw an E-R diagram.

Solution:

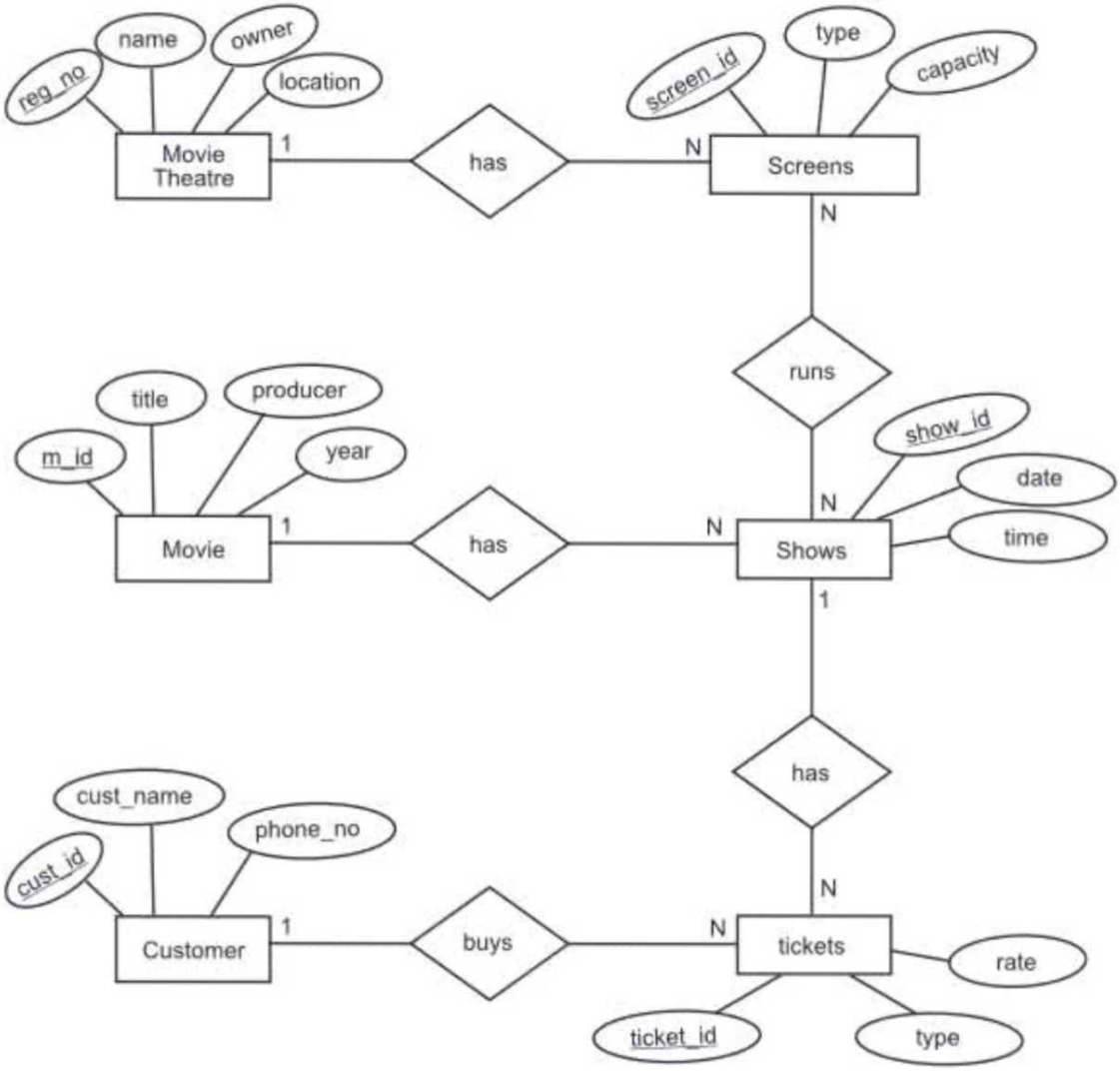


Fig. 3.25

8. A publisher publishes many books. A book can be written by several authors and an author can write several books. For a course there are several books. Draw an E-R diagram.

Solution:

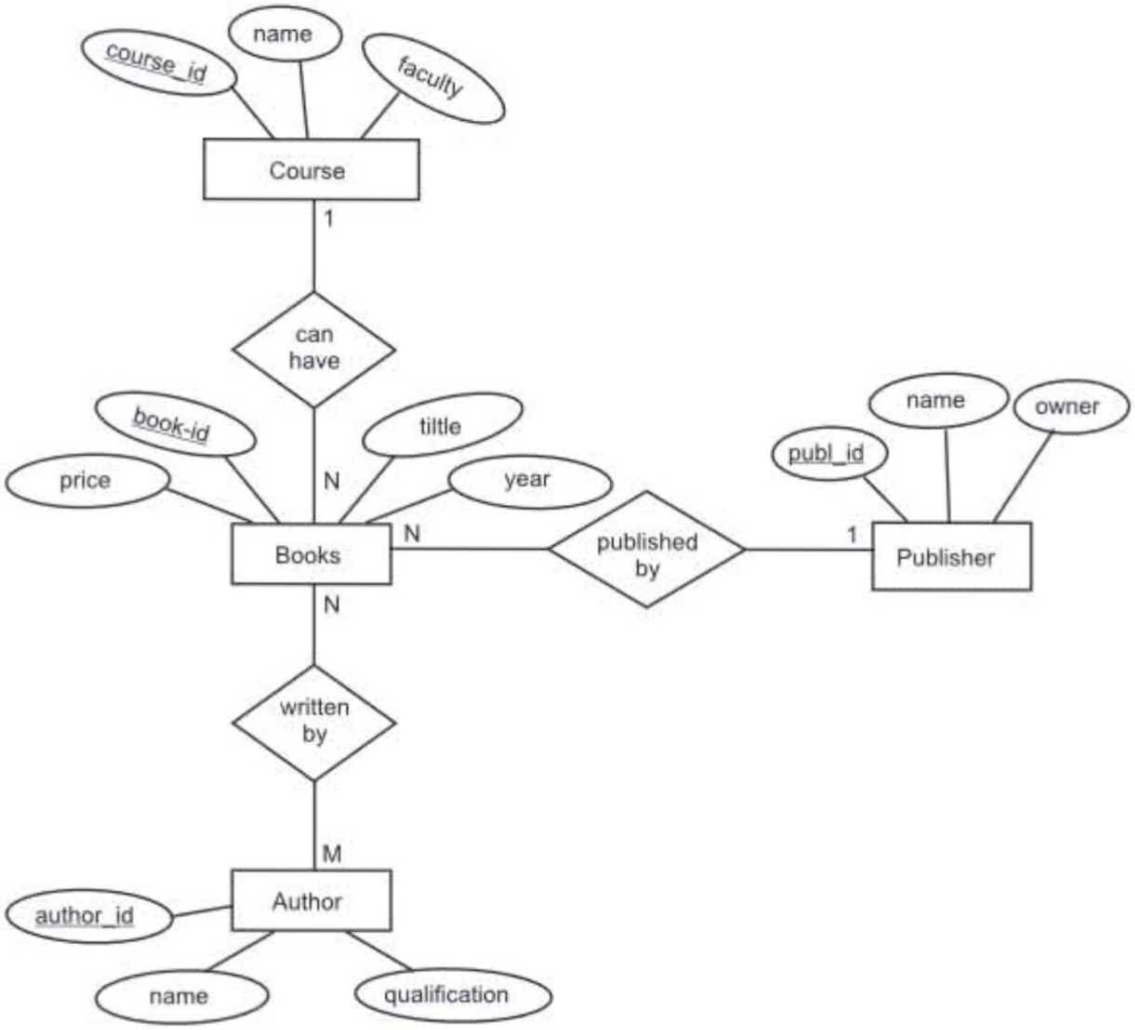


Fig. 3.26

---

9. A transport company has several trucks. Each truck carries many shipments. The shipments are delivered to a warehouse. Every warehouse has several stores linked with it. Draw an E-R diagram.

Solution:

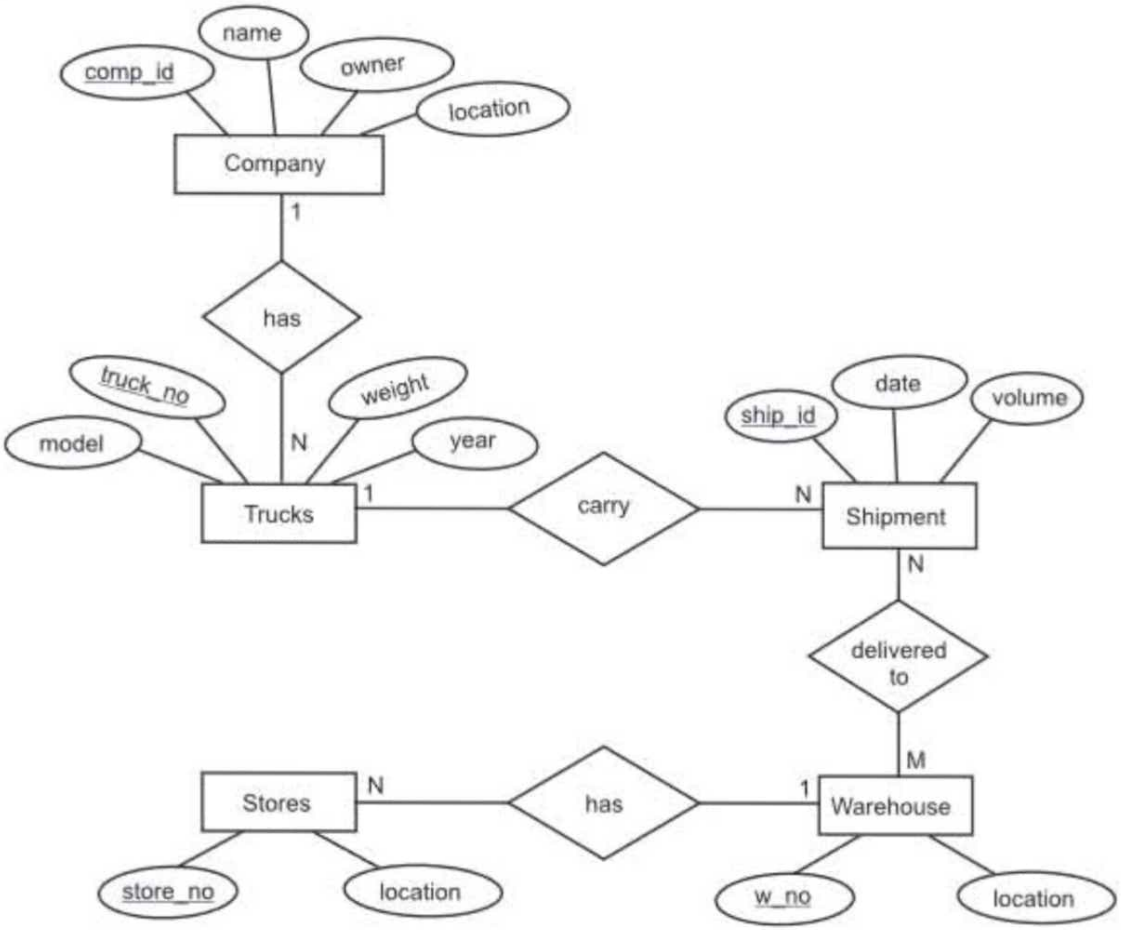


Fig. 3.27

Summary

- The main objectives of database designing are to produce logical and physical design model for the new proposed system.
- E-R model was developed by Peter Chen. It is based on the concept of real-world entities and the relationship between them.
- An entity is an object that is distinguishable from other.
- An entity set is a set of all entities of the same type.
- A member of a strong entity set is called dominant entity, whereas a member of weak entity set is called as subordinate entity.
- In case of weak entity set, total participation in the identifying relationship always exists.

- An entity can be described with the help of some properties, known as attributes of an entity.
- Types of attributes: Simple, Composite, Single-valued, Multi-valued, Stored, Derived, Null.
- A relationship is an association among two or more entities.
- These are three types of degrees of relationships: unary, binary and ternary.
- Mapping cardinalities describe the number of entities to which another entity can be linked via a relationship set.
- E-R diagram can be drawn by using notations like Chen notation, Crow's foot notation. The Crow's foot notation was invented by Gordon Everest.
- A key is a set of attributes that uniquely identifies an entity within an entity set.
- In total participation, every entity in an entity set E is involved in the relationship.
- Specialization is the process of identifying subsets of an entity that share some distinguishing characteristics.
- Generalization is the process of identifying common characteristics among the lower-level entity set to synthesize into a single, higher-level entity set.
- Generalization works in bottom up manner while specialization works in top down manner.
- Aggregation is an abstraction through which relationship is treated as higher level entity set.

Check Your Understanding

Q.1. Multiple Choice Questions:

1. Entity relationship model is widely used in .....
   (a) database implementation (b) database direction
   (c) database processing (d) database design
2. The relationship set is represented in E-R diagram as .....
   (a) diamond (b) double diamonds
   (c) dashed lines (d) rectangles
3. An entity set that does not have sufficient attributes to form a primary key is known as .....
   (a) Strong entity set (b) weak entity set
   (c) both of these (d) none of these
4. Weak entity set contains a .....
   (a) primary key (b) foreign key
   (c) discriminator (d) none of these

5. ..... attribute can be splitted into meaningful subparts.
   (a) simple (b) composite
   (c) stored (d) derived
6. Multi valued attributes are represented by ..... ellipse.
   (a) dashed (b) double
   (c) single (d) none of these
7. Derived attribute is represented by ..... ellipse.
   (a) dashed (b) double
   (c) single (d) none of these
8. A relationship that involves one entity is called ..... relationship.
   (a) single (b) unary
   (c) binary (d) ternary
9. The Crow's foot notaion was invented by ..... .
   (a) Gordon Everest (b) Peter Chen
   (c) E.F. Codd (d) none of these
10. E-R model was developed by ..... .
    (a) Peter Chen (b) E.F. Codd
    (c) Gordon Everest (d) none of these
11. ..... is a top-down process.
    (a) Specialization (b) Generalization
    (c) Aggregation (d) none of these
12. ..... is a bottom-up process.
    (a) Specilization (b) Generalization
    (c) Aggregation (d) None of these
13. ..... is an abstraction through which relationship is treated as higher level-entity set.
    (a) Specilization (b) Generalization
    (c) Aggregation (d) None of these

Ans.: (1) - d (2) - a (3) - b (4) - c
(5) - b (6) - b (7) - a (8) - b
(9) - a (10) - a (11) - a (12) - b
(13) - c

Q.2. State True/False.

1. There is no role of the needs of the users in database design process.
2. In logical design stage, the designer maps the high-level conceptual schema onto the implementation data model of the DBMS.
3. It is recommended to complete E-R modelling before implementing the database.
4. An entity set is a set of entities of different types.
5. Strong entity does not have a primary key.
6. In case of weak entity set total participation in the identifying relationship always exists.
7. Domain is a set of all possible values that an attribute can have.
8. Birth-date is a derived attribute.
9. Null value means missing or unknown.
10. A unary relationship is also called recursive.
11. An identifying relationship is represented by double diamond.
12. Total participation is represented by dashed lines.
13. Candidate key is the minimal super key.

Ans.: (1) False (2) True (3) True (4) False
(5) False (6) True (7) True (8) False
(9) True (10) True (11) True (12) False
(12) True

Practice Questions

Q.1. Answer the following questions in brief.

1. What are the phases of database design ?
2. What is E-R modelling ?
3. What are the different types of entity set ?
4. State the different types of attributes.
5. What is a null attribute ?
6. What is ternary relationship ?
7. Differentiate between total and partial participation.
8. What is modality of the relationship ?
9. Discuss in brief about four possible edges to the relationship in Crow's foot notation.
10. What is a superkey ?

Q.2. Answer the following questions.

1. Explain the suitable diagram the phases of database design.
2. Differentiate between strong and weak entity set.

3. Explain in detail about types of attributes.
4. Explain mapping cardinalities in detail.
5. Compare between cardinality and modality.
6. What are key constraints ?
7. Differentiate between specialization and generalization.
8. What is aggregation ?
9. Explain in detail about types of attributes.

Q.3. Define the terms.

(a) Entity
(b) Entity set
(c) Weak entity
(d) Strong entity
(e) Composite attribute
(f) Derived attribute
(g) Null attribute
(h) Relationship
(i) Degree of a relationship type
(j) Primary key
(k) Generalization

Previous Exams Questions

Summer 2017

1. A key consisting of two or more columns is called as ..... . **[1 M]**

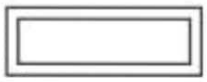
(i) Composite key
(ii) Candidate key
(iii) Key
(iv) Database key

Ans.: Refer to Section 3.3.1.

2. Which of the following notations represents weak entity ? **[1 M]**

(i) 

(ii) 

(iii) 

(iv) 

Ans.: Refer to Section 3.2.1.

3. Differentiate between strong entity and weak entity. **[5 M]**

Ans.: Refer to Section 3.2.1.

4. Write a short note on participation constraints in ER model. **[3 M]**

Ans.: Refer to Section 3.3.

5. An IT industry is developing several projects on various domains (banking, education, inventory etc.) for many of its clients. Many IT professionals are working on one project and an IT professional can work on many projects:
   (i) Identify the entity sets, their attributes and primary key for each entity.
   (ii) Identify the relationship set and draw and ER diagram. **[4 M]**

Ans.: Refer to Section 3.3.4.

6. Write a note on 'Generalization' with Example. **[3 M]**

Ans.: Refer to Section 3.3.3.

7. Consider the relations:
   donor(did, dname, addr)
   patient(pid, pname, paddr.)
   donor and patient are related by many to many with descriptive attribute date of donation:
   (i) Draw an ER diagram **[3 M]**

Ans.: Refer to Section 3.3.4.

(ii) Normalize Database with necessary constrains.

Ans.: Refer to Chapter 6 Section 6.8.

8. What is Relationship Set. **[1 M]**

Ans.: Refer to Section 3.3.3.

Winter 2017

1. In entity-relationship diagram double ellipse represents ..... . **[1 M]**
   (i) multivalued attribute (ii) derived attribute
   (iii) weak entity (iv) primary key

Ans.: Refer to Section 3.2.1.

2. Process of breaking a large relation R into a set of small r_1, r_2, ... r_n is called as ..... . **[1 M]**
   (i) association (ii) generalization
   (iii) decomposition (d) none of the above

Ans.: Refer to Section 3.3.3.

3. What do you mean by strong and weak entity sets ? **[1 M]**

Ans.: Refer to Section 3.2.1

4. Explain any three types of attributes of entity relationship model in detail. **[3 M]**

Ans.: Refer to Section 3.2.2.

5. Consider the following relations:

   Book (bno, bname, publication, price)

   Author (ano, aname, address)

   Book and Author are related with many to many relationship. Draw an ER diagram for above scenario. **[3 M]**

Ans.: Refer to Section 3.3.4.

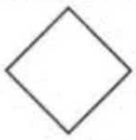
6. Consider the following Relational Database 'Star' in an agency for flate booking and it has number of builders and agents who are jointly working. A customer can get a flat for residential or commercial purpose. If customer is approached through an agent, the agency and builders are giving some commission to the agent. Agent shows various flats and sites within various locations. Study above case and:

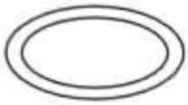
   (i) Design and ER diagram

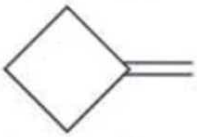
   (ii) Identify all entities. **[4 M]**

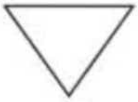
Ans.: Refer to Section 3.3.4.

7. Explain any three ER notations with example:

   (i) 

   (ii) 

   (iii) 

   (iv) 

Ans.: Refer to Section 3.2.1.

Summer 2018

1. In ER diagram, weak entity is represented by ..... . **[1 M]**

   (i) double rectangle (ii) Rectangle

   (iii) double ellipse (iv) dashed ellipes

Ans.: Refer to Section 3.2.1.

2. Define: weak entity set [1 M]

Ans.: Refer to Section 3.2.1.

3. A motor vehicle branch administers driving tests and issues driver's licenses. Any person who wants a driver's licence must first take a learner's exam at any motor vehicle branch in the province. If he/she fails the exam, he can take the exam again any time after a week of the fialed exam date, at any branch. If he passes the exam, he is issued a license (type=learners) with a unique licence number. The person may take his drives exam at any branch any time before the learner's licence expiry date (Which is usually set at six months after the licence issue data). If he passes the exam, branch issues him a driver's licence. Identify the entities and attributes. Draw ER diagram. [5 M]

Ans.: Refer to Section 3.3.4.

4. Write a short note on key constraints. [5 M]

Ans.: Refer to Section 3.3.1.

5. Consider following relational schema.
   Game (ano, gname, no. of players, coach name, captain)
   Player (Pno, Pname).
   - Game and player are related with many-many.
   - Draw an ER diagram. [3 M]

Ans.: Refer to Section 3.3.4.

4...

Structure of Relational Databases

Objectives...

- To understand basic concepts in relational database.
- To have knowledge about conversion of E-R to relational model.
- To understand the various mappings in conversion of E-R to relational model.
- To know the integrity constraints and its meaning.

4.1 INTRODUCTION (S-17)

- A database is a collection of data that is organized. Database support storage and manipulation of data.
- Database are based on one of the four models, i.e. the hierarchical model, the network model, the relational model, the object oriented relational model.
- A database based on the network or hierarchical model is a non-relational database. A database based on the relational model is a relational database.
- Relational Database Management System (RDBMS) is the most popular DBMS type.

4.2 CONCEPTS (S-17, 18, W-17)

(i) **Table:** A table is a two-dimensional structure. A table has rows (horizontal dimension) and columns (vertical dimension). Rows in a table represent records (tuples) and columns represent the attributes (fields) in a relational database.

(ii) **Tuple (row):** A single row of a table which contains a single record representing a set of related data. Tuple is a horizontal dimension in a table.

(iii) **Column:** A column is a set of values for a specific attribute. Column is a vertical dimension in a table.

(iv) **Attribute:** It is a named column of a relation (table).

(v) **Domain:** It is a set of possible values of one or more attributes. For example, the domain for roll number attribute is a set of all non-negative numbers.

(vi) **Degree:** The degree of a relation is the number of attributes contained in a relation. For example, if a student relation is having roll_number, name, class, division, address fields then the degree of student relation is 5.

(vii) **Cardinality:** The cardinality of a relation is the number of tuples (rows) present in a relation (table).

(viii) **Relation schema:** A relation schema describes the exact structure of the relation. It describes the table (relation) name, the attributes with their names.

(ix) **Relation key:** One or more attributes which uniquely identify a row in the relation (table) is called the relation key.

(x) **Relation instance:** It is a finite set of tuples in the relational database system. Relation instances are mutually exclusive, which means that they do not have duplicate tuples (records).

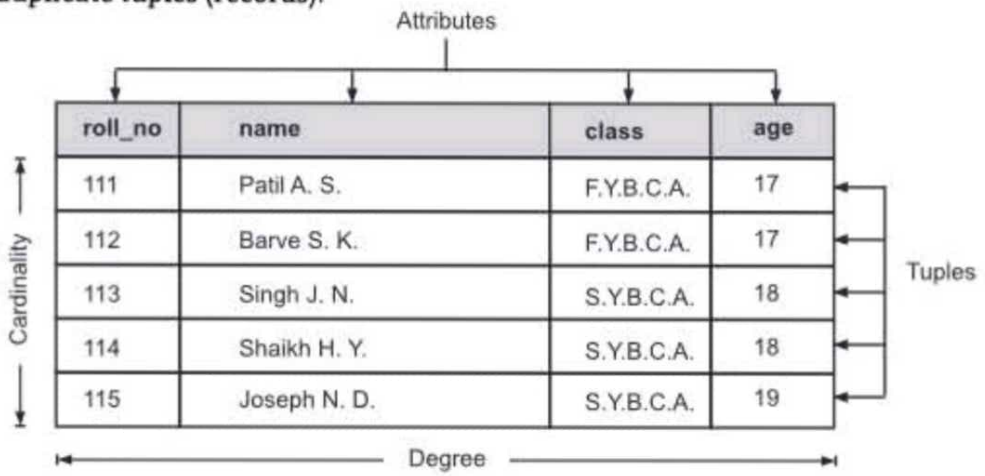


Fig. 4.1: Elements of Relational Database Structure

- In the fig. 4.1 the degree of table is 4 and cardinality of table is 5.
- The domain of roll no. is 101 to 200. The domain of name is any noun containing combination of alphabets.
- The domain of age is 17 to 25. The domain of class is {F.Y.B.C.A., S.Y.B.C.A., T.Y.B.C.A.}

4.3 CONVERSION OF ER TO RELATIONAL MODEL (S-17, 18, W-17)

- In conversion of E-R model to relational model, each entity set and each relationship set is converted to a relation.

Types of Keys in RDBMS:

1. **Primary Key:** The attribute or combination of attributes which uniquely identify each row (tuple) of a relation (table) is called a Primary Key. Primary key cannot contain duplicate or null values.

Primary key can be simple or composite. "When primary key is a combination of two or more attributes, it is called as **Composite key**".

2. **Unique Key:** An attribute (column) which contains unique (non-repeated) values is known as Unique Key. Unique key can accept only one NULL value for column.
3. **Foreign Key:** A foreign key is a column or group of columns in a table that uniquely identifies a tuple (row) in another table. It acts as a cross-reference between tables. It references the primary key of another table. The table containing primary key is called the *parent table* and the table containing foreign key is called *child table.*

- We can have a foreign key that references a unique key of another table. A table can have only one primary key but there can be many unique keys defined on a table.

4.3.1 Mapping of Strong Entity Sets

- There is a direct mapping between the E-R model and the relational model.
- Create table for each strong entity. The attributes of entity are represented as fields (columns) of table.
- The attribute or set of attributes that uniquely identify a tuple (row) are chosen as primary key.

  Single attribute: Simple primary key

  Set of attributes: Composite primary key

For example, consider the entity set *loan*. It has three attributes loan_no, sanction_date, amount. We represent this entity set by loan table. The loan_no attribute is the primary key.

loan_no	sanction_date	Amount (₹)
10	10/07/2019	50000
11	15/08/2019	70000
12	20/08/2019	100000
13	25/09/2019	150000
14	25/09/2019	170000

Fig. 4.2: loan Table

4.3.2 Mapping of Weak Entity Sets

- Let R be a weak entity set and S be the strong entity set on which R depends. Create a table containing all attributes of weak entity set.
- Additionally include the primary key of the strong entity set in the table containing all attributes of weak entity set as a foreign key.
- Create a primary key for this table by combining the primary key of strong entity set and the discriminator of the weak entity set.

For example, consider the entity set *installment*. This entity set has three attributes: installment number, installment date, installment amount. The primary key of the loan entity set is loan-no. Thus we create installment table with four attributes:
installment (loan_no, installment_no, intallment_date, installment_amount)
Primary key: {loan_no, installment_no} (composite)
Foreign key: {loan_no}

4.3.3 Mapping of Relationship Sets

- A Mapping cardinality is a data constraint that specifies how many entities an entity can be related to ain a relationship set.
- A binary relationship set is a relationship set on two entity sets.

4.3.3.1 Binary One-to-One Relationship

- For a binary one-to-one relationship between S and T, choose one of the relations 'S' and include the primary key of 'T' in 'S' as the foreign key.
- Choose an entity with total participation in relationship in the role of S.
- For example, there is a one-to-one relationship between patient and bed. The participation of bed in the assigned_to relationship is total. We include the primary key of the patient as a foreign key in the bed table.

4.3.3.2 Binary One-to-Many Relationship

- For a binary one-to-many relationship, identify the relation at the many side of the relationship.
- Include the primary key of the relation at the one side into the relation at the many side as a foreign key.
- For example, department and employee are related with one-to-many relationship. The primary key of department is included in the employee table as a foreign key.

4.3.3.3 Binary Many-to-Many Relationship

- For binary many-to-many relationship, create a new relation containing the primary keys of both the relations. Here the primary keys of both relations act as foreign keys in the third table.
- The combination of the foreign keys forms the primary key (composite) of the third table.
- For example, student and subject are related with many-to-many relationship. In the third table, roll number and subject-number are added as foreign keys. Roll number and subject-number forms the primary key (composite) of the third table.

4.3.4 Mapping of Generalization or Specialization

- There are two different methods of designing tables for an E-R diagram showing Generalization or Specialization.
- Consider that doctor is a higher level entity set.
- There are two lower-level entity sets namely permanent and visiting. Let us consider that registration_no is the primary key of doctor.
  1. Create a table for the higher-level entity set. Create a table for each lower-level-entity set that includes all attributes of that entity set plus one for each attribute of the primary key of the higher-level entity set.
     Thus, the table design is as follows:
     doctor (registration_no, name, street, city)
     permanent (registration_no, salary)
     visiting (registration_no, consultation_fee)
     The primary key attributes are underlined. We create foreign key referencing the primary key of the relation created from the higher-level entity set. In the above example, the registration_no attribute of permanent and visiting reference the primary key of doctor.
  2. Do not create a table for the higher-level entity set. For each lower-level entity set, create a table that includes all attributes of that entity set plus all attributes of the higher-level entity set.
     Then the table design is as follows:
     permanent (registration_no,name, street, city, salary)
     visiting (registration_no, name, street, city, consultation fee)
     Both of these tables have registration_no as the primary key attribute. It is the primary key attribute of the higher-level entity set doctor.

4.3.5 Composite and Multivalued Attributes

- Composite attributes are handled by creating a separate attribute for each of the component attributes.
  For example, name is a composite attribute of student entity set. The components of name are first name, middle name and last name. The table student contains attributes first name, middle name, last name.
- For a multivalued attribute, we create a table (relation) with the primary key of the entity set and an attribute that corresponds to the multivalued attribute.
  For example, the student entity set has a multivalued attribute contact_number. For this, we create a table contact_info.
  contact_info (roll_no, contact_no)
  In the above example, primary key is underlined. The roll_no is a foreign key that references primary key roll_no of student table.

4.4 INTEGRITY CONSTARINTS (W-17, S-18)

- The constraints are used to check the modification or insertion of a data, are called integrity constrains.
- While designing a relational data model, the database designer must be aware of all the integrity constraints that need to be satisfied.

4.4.1 Entity Integrity

- Entity integrity constraint states that primary key value can be never NULL.
- The concept of this constraint is related with the concept of primary key.
- This ensures that values in the primary key column are not missing or undefined. It is also known as **Integrity Rule 1**.

roll_no	name	class	division
10	Aniket	1	A
11	Mandar	7	B
11	Krishna	2	A
	Chinmay	5	A

Not allowed because roll_no is a primary key

Fig. 4.3: Entity Integrity

4.4.2 Referential Integrity

- Referential Integrity constraint is specified between two tables (i.e. parent and child). It maintains the consistency among tuples in the two tables.
- It states that "a tuple in one table that refers to another table (relation) should refer to an existing tuple in that table (relation)".

Department Table

Primary key →

Dept_no	Dept_name
1	Production
2	QA
3	Engineering

Employee Table

emp_no	emp_name	Age	dept_no	← foreign key
10	Anil	30	1	
11	Ramesh	28	1	
12	Tanvir	32	2	
13	Reema	26	2	
14	Ravi	28	4	← not allowed as dept_no 4 is not present in parent table

Fig. 4.4: Referential Integrity

- Referential Integrity constraint is also known as **Integrity Rule 2**.

4.4.3 Null Constraint

- Null means missing or unknown value. A null value in a column of a table indicates that the value for that column is unknown.
- The *not null* specification does not allow the insertion of a null value for this specified column (attribute). We want to avoid null values in many situations.

4.4.4 Unique Constraint

- Unique constraint ensures that a field or a column will only have unique (non-repeated) values. A unique constraint field (column) will not have duplicate data.
- The unique (A_1, A_2 . . . . A_m) specification says that attributes A_1, A_2 . . . . A_m form a candidate key i.e. no two tuples in the relation agree on all the primary-key attributes. But candidate key attributes are allowed to have null value unless they have been declared explicitly to be *not null*.

4.4.5 Check constraint

- The check constraint is used to restrict the value of a column within a range. It performs check on the values, before they are stored into the database. We can use the check clause to simulate an enumerated type. Complex check conditions may be useful when we want to ensure integrity of data. But we must use them wisely, because we may find them costly to test.

4.4.6 Examples of E-R Diagram to Relational Model (S-17, W-17, S-18)

Example 1: Convert the following E-R diagram into relational model.

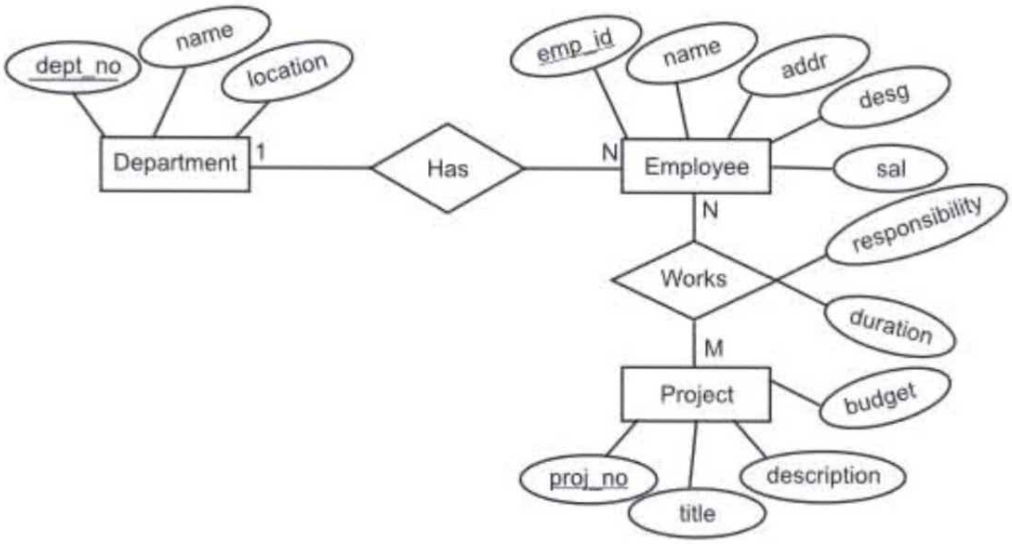


Fig. 4.5

Solution:

Table Name : Employee
- emp_id (Primary Key)
- name
- addr
- desg
- sal
- dept_no (Foreign Key)

Table Name : Department
- dept_no (Primary Key)
- name
- location

Table Name : Project
- proj_no (Primary Key)
- title
- description
- budget

Table Name : Works
- proj_no (Foreign Key)
- emp_id (Foreign Key)
- responsibility
- duration
- composite primary key : (proj_no, emp_no)

Example 2: Convert the following E-R diagram into relational model.

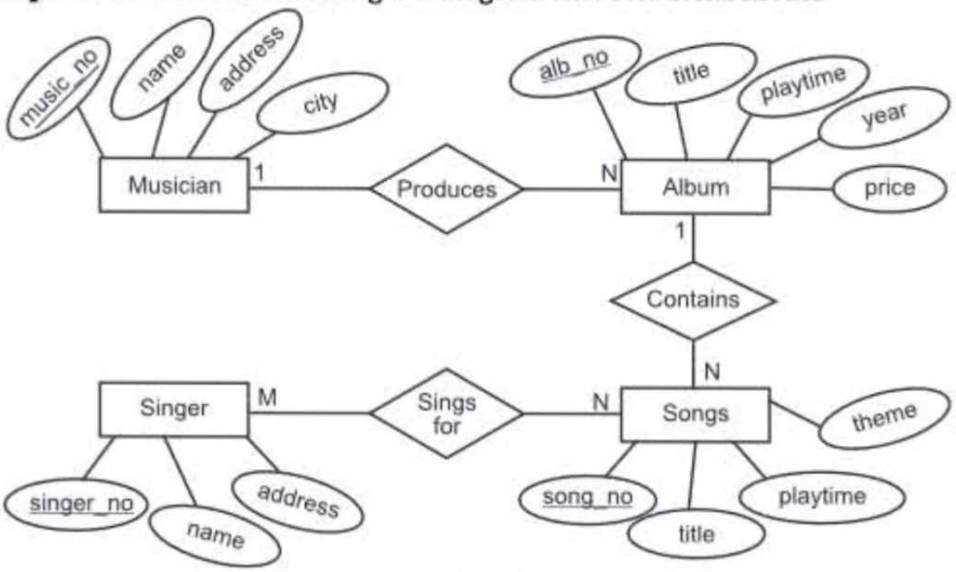


Fig. 4.6

Solution:

Table Name : Album
- alb_no (Primary Key)
- title
- playtime
- year
- price
- music_no (Foreign Key)

Table Name : Musician
- music_no (Primary Key)
- name
- address
- city

Table Name : Singer
- singer_no (Primary Key)
- name
- address

Table Name : Songs
- song_no (Primary Key)
- title
- playtime
- theme

alb_no (Foreign Key)

Table Name : Singsfor

song_no (Foreign Key)

singer_no (Foreign Key)

composite primary key : (song_no, singer_no)

Example 3: Convert the following E-R diagram into relational model.

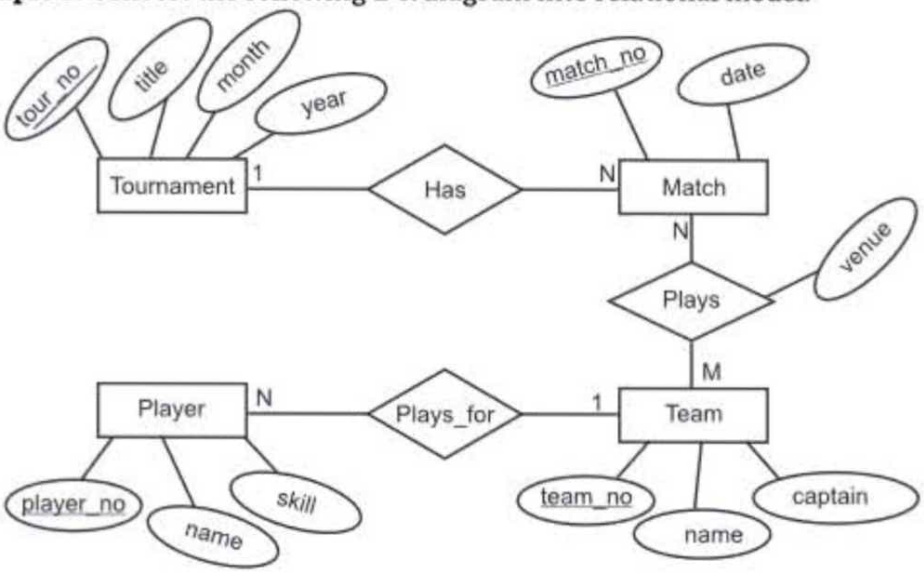


Fig. 4.7

Solution:

Table Name : Tournament

tour_no (Primary Key)

title

month

year

Table Name : Match

match_no (Primary Key)

date

tour_no (Foreign Key)

Table Name : Team

team_no (Primary Key)

name

captain

Table Name : Plays

team_no (Foreign Key)

match_no (Foreign Key)

venue
composite primary key : (team_no, match_no)

Table Name : Player
player_no (Primary Key)
name
skill
team_no (Foreign Key)

Example 4: Convert the following E-R diagram into relational model.

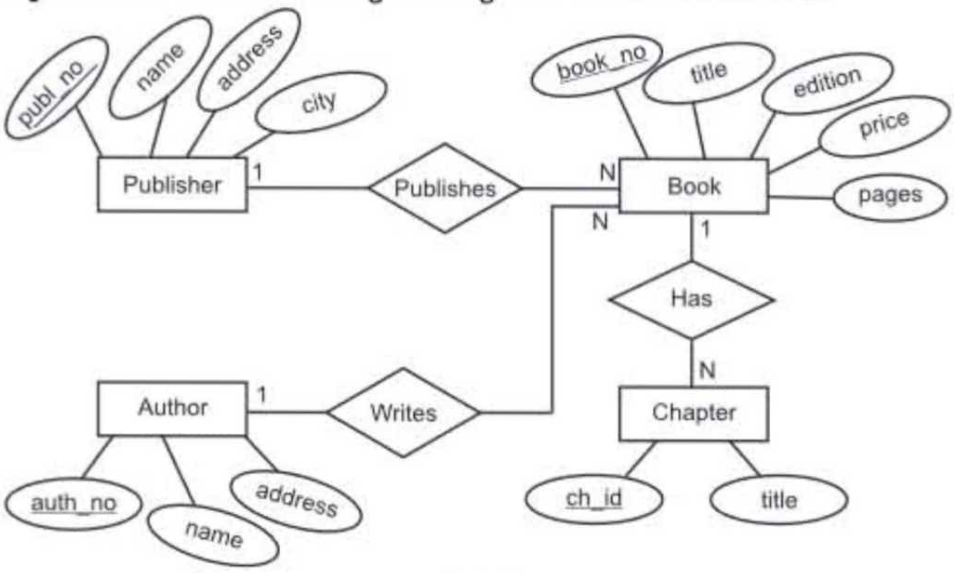


Fig. 4.8

Solution:

Table Name : Publisher
publ_no (Primary Key)
name
address
city

Table Name : Book
book_no (Primary Key)
title
edition
price
pages
publ_no (Foreign Key)
auth_id (Foreign Key)

Table Name : Author
auth_id (Primary Key)
name
address

Table Name : Chapter
ch_id (Primary Key)
title
book_no (Foreign Key)

Summary

- A database is a collection of data that is organized. It supports storage and manipulation of data.
- A table is a two-dimensional structure.
- Column is vertical dimension and tuple is horizontal dimension in a table.
- Domain is a set of possible values for one or more attributes.
- Degree of a relation = number of attributes contained in a relation.
- Cardinality of a relation = number of tuples present in a relation.
- Relation instances do not have duplicate tuples.
- Primary key uniquely identifies each row of a relation.
- When primary key is a combination of two more at tributes, it is called as composite key.
- Unique key can accept only one NULL value for column.
- A foreign key references the primary key of another table. It acts as a cross-reference between tables.
  Table containing primary key = parent table
  Table containing foreign key = child table
- A table can have only one primary key but there can be many unique keys defined on a table.
- Primary key for a table of weak entity set = primary key of strong entity set + discriminator of weak entity set.
- For a binary many-to-many relationship create a new relation containing the primary keys of both the relations.
- Composite attributes are handled by creating a separate attribute for each of the component attributes.
- Entity Integrity ensures that values in the primary key column are not missing or undefined (Integrity Rule 1).
- Referential Integrity maintains the consistency among tuples in the two tables. (Integrity Rule 2)
- Null value means missing or unknown value.
- A unique constraint field will not have duplicate data.
- The check constraint is used to restrict the value of a column within a range.

Check Your Understanding

Q.1. Multiple Choice Questions:

1. A table is a ..... dimensional structure.
   (a) two (b) three
   (c) four (d) none of these
2. ..... is a set of possible values for one or more attributes.
   (a) Attribute (b) Domain
   (c) Column (d) none of these
3. The ..... of a relation is the number of attributes contained in a relation.
   (a) Cardinality (b) degree
   (c) size (d) none of these
4. ..... is a finite set of tuples in the relational database system.
   (a) Relation key (b) Relation instance
   (c) Relation schema (d) none of these
5. ..... key cannot contain duplicate or null values.
   (a) Primary (b) Unique
   (c) Foreign (d) none of these
6. ..... key can accept only one NULL value for column.
   (a) Primary (b) Foreign
   (c) Unique (d) none of these
7. The table containing primary key is called the ..... table.
   (a) child (b) parent
   (c) detail (d) none of these
8. There can be many ..... keys defined on a table.
   (a) unique (b) primary
   (c) both unique and primary (d) none of these
9. For a one-to-many relationship, include the primary key of the relation at the ..... side into the relation at the ..... side.
   (a) one, many (b) many, one
   (c) one, one (d) many, many
10. For a ..... relationship, create a new relation containing the primary keys of both the relations.
    (a) one-to-one (b) one-to-many
    (c) many-to-many (d) none of these

11. ..... integrity is also known as Integrity Rule 1.
    (a) Referential (b) Entity
    (c) Table (d) none of these

Ans : 1-a, 2-b, 3-b, 4-b, 5-a, 6-c, 7-b, 8-a, 9-a, 10-c, 11-b.

Q. 2. State True/False.

1. A database based on the network or hierarchical model is a non-relational database.
2. Tuple is a vertical dimension in a table.
3. Attribute is a named column of a relation.
4. The cardinality of a relation is the number of tuples present in a relation.
5. Relation instances do not have duplicate tuples.
6. Primary key can be simple or composite.
7. Unique key cannot accept NULL value.
8. A table can have only one primary key but there can be many unique keys defined on a table.
9. There are two different methods of designing tables for an E-R diagram showing – generalization or specialization.
10. Composite attributes are handled by creating a separate attribute for each of the component attributes.
11. Referential Integrity constraint is specified between parent and child tables.

Ans.: (1) True (2) False (3) True (4) True
(5) True (6) True (7) False (8) True
(9) True (10) True (11) True

Practice Questions

Q.1. Answer the following questions in brief.

1. What is a relation instance ?
2. What is a foreign key ?
3. State the difference between strong entity set and weak entity set.
4. Explain the conversion of weak entity set in a table.
5. Explain the conversion of many-to-many relationship in a table.
6. How do deal with composite attribute in table design from an E-R diagram ?
7. What is entity integrity ?
8. What is referential integrity ?
9. What is null constraint ?
10. What is unique constraint ?
11. State the use of check constraint.

Q.2. Answer the following questions.

1. With a suitable diagram explain the elements of relational database structure.
2. Explain mapping of relationship set for a binary one-to-one relationship.
3. Explain mapping of relationship set for a binary one-to-many relationship.
4. Explain mapping of generalization or specialization.
5. How to deal with multivalued attribute in the table design from an E-R diagram.
6. Explain with suitable diagram how entity integrity is maintained ?
7. Explain with suitable diagram how referential integrity is maintained ?
8. Compare between primary key and unique key.
9. Compare between primary key and foreign key.
10. Differentiate between entity integrity and referential integrity.
11. Explain in detail about mapping of weak and strong entity sets.

Q.3. Define the terms.

(a) Table
(b) Tuple
(c) Domain
(d) Relation schema
(e) Primary key
(f) Unique key
(g) Column
(h) Attribute
(i) Degree
(j) Cardinality
(k) Relation key

Previous Exams Questions

Summer 2017

1. The attributes which act as primary key in one table and a reference key in another table is a ..... . **[1 M]**
   (i) Primary key
   (ii) Super key
   (iii) Foreign key
   (iv) None of the above

Ans.: Refer to Section 4.2.

2. Define the term 'Cardinality'. **[1 M]**

Ans.: Refer to Section 4.1.

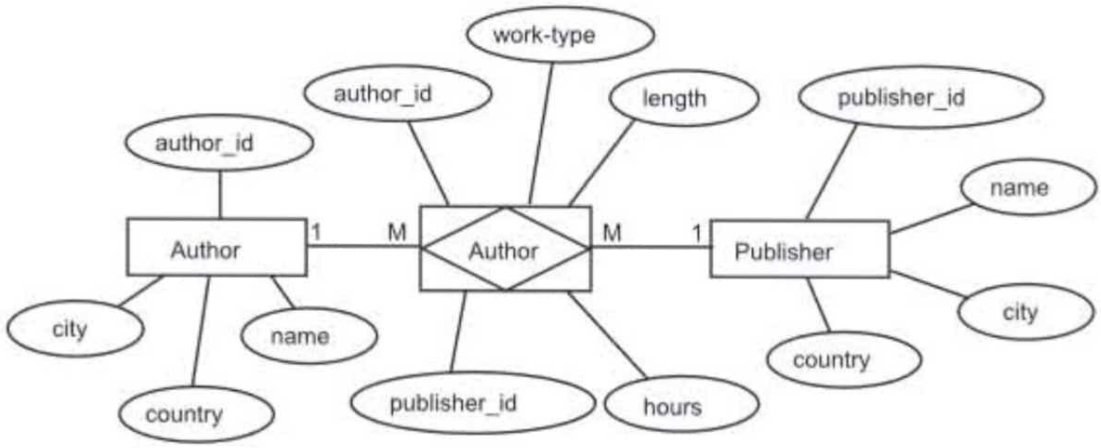
3. In ..... database we have strict parent-child relationship only. **[1 M]**
   (i) network model
   (ii) hierachical model
   (iii) relational model
   (iv) none of the above

Ans.: Refer to Section 4.2.

4. Discuss various types of keys used in RDB. **[4 M]**

Ans.: Refer to Section 4.2.

5. Consider the ER diagram given 'below and convert it into relational model. **[3 M]**



Ans.: Refer to Section 4.4.6.

Winter 2017

1. Relational data model stores the data in the form of ..... . **[1 M]**
   (i) row
   (ii) column
   (iii) relation
   (iv) table

Ans.: Refer to Section 4.2.

2. Define the term 'tuple'. **[1 M]**

Ans.: Refer to Section 4.2.

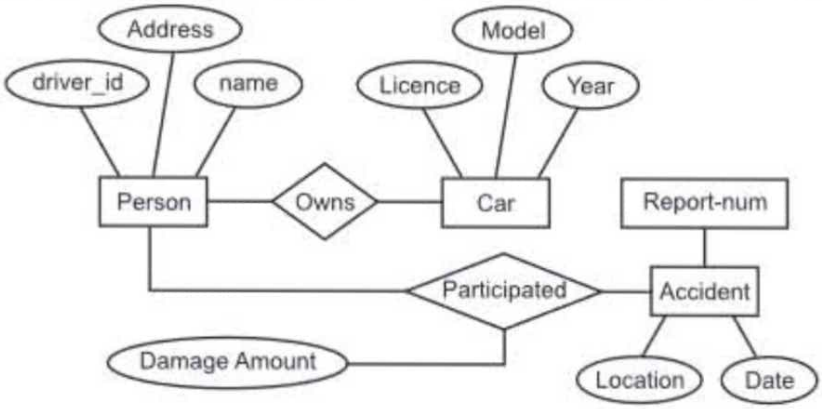
3. Define Key. Explain the following terms : **[5 M]**
   (i) Primary key
   (ii) Foreign key
   (iii) Super key
   (iv) Candidate key

Ans.: Refer to Section 4.2.

4. Write a short note on integrity constraints on database design. **[4 M]**

Ans.: Refer to Section 4.3.

5. Design a relational database corresponding to the following ER diagram.



Ans.: Refer to Section 4.4.6.

Summer 2018

1. In relational model, relations are termed as ..... . **[1 M]**

   (i) tuples (ii) attributes

   (iii) tables (iv) rows

Ans.: Refer to Section 4.2.

2. What do you mean by a domain ? **[1 M]**

Ans.: Refer to Section 4.2.

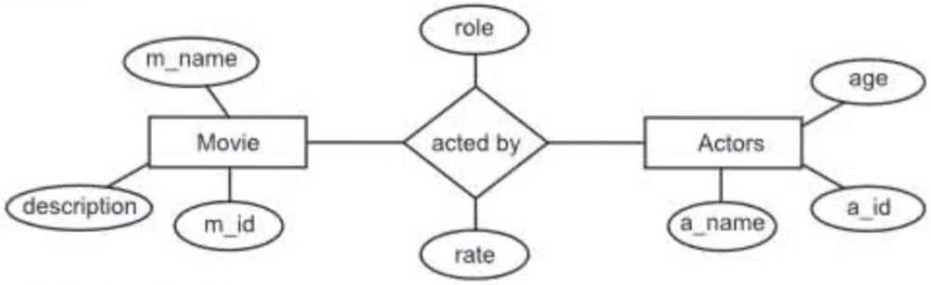
3. Define constraint state and explain any two types of constraint. **[4 M]**

Ans.: Refer to Section 4.3.

4. Define a key. Differentiate between primary key, candidate key and super key.

Ans.: Refer to Section 4.2.

5. Consider following entity relationship diagram and convert it into relational model. **[3 M]**



Ans.: Refer to Section 4.4.6.

5...

SQL

Objectives...

- To know about DDL and DML commands.
- To learn how to write simple and nested queries.
- To know the importance of SQL in RDBMS.

5.1 INTRODUCTION

- SQL stands for Structure Query Language. It is accepted as a standard language for accessing and manipulating databases. All the relational database management systems like MySQL, MS-Access, Oracle, Postgres, Sybase, Informix, SQL Server use SQL as their standard database language. SQL was developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s. The initial version was called SEQUEL(Structured English Query Language). The term "SEQUEL" was later changed to "SQL".

Features of SQL:

(i) SQL is very flexible.

(ii) It is used to describe the data.

(iii) SQL is very simple and easy to learn (SQL statements are simple English sentences).

(iv) SQL can be embedded in front end application code like Visual Basic, Java and so on.

(v) SQL is a high level language.

(vi) SQL based applications are portable across computer systems, which means they can be run any system.

Architecture of SQL

- Fig. 5.1 shows SQL architecture which gives us a detailed idea about the process.

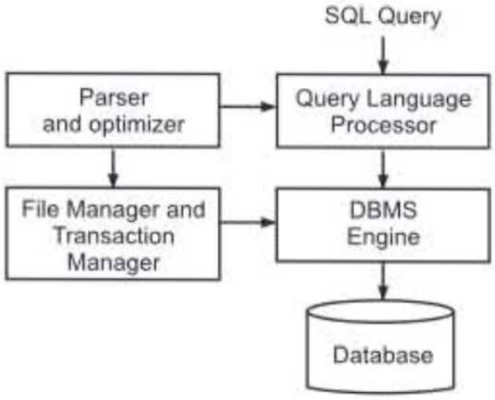


Fig. 5.1: SQL Architecture

5.1.1 Advantages of SQL

(i) Using SQL one can manage database systems easily without the need to write large amount of code.

(ii) SQL is portable. SQL supports PCs, Laptop, Tablets, Servers, Mainframes. We can easily move the database using SQL from one device to another.

(iii) Using SQL the users can create different views of the database for the different type of users.

(iv) Using SQL, we can obtain answers to complex queries in few seconds. We can write ad hoc queries to access the part of database.

(v) SQL is an accepted standard. It is being used by ANSI(American National Standard Institutes) and ISO(International Organization for Standardization).

5.2 QUERY PROCESSING STEPS

- Query processing includes translation of high-level queries into low-level expressions that can be used at the physical level of the file system, query optimization and actual execution of the query to get the result.
- It is a three-step process that consists of parsing and translation, optimization and execution of the query submitted by the user as shown in Fig. 5.2.

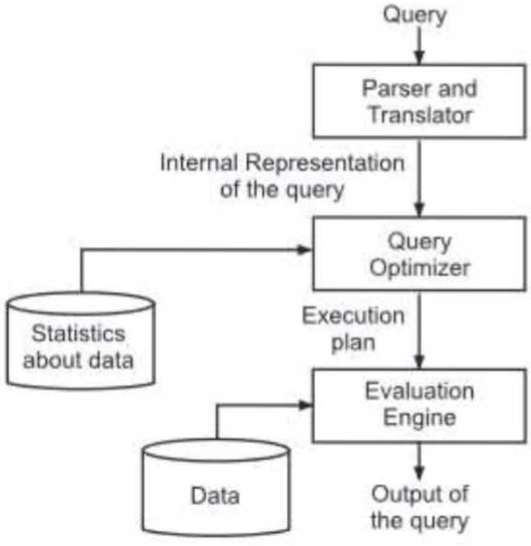


Fig. 5.2: Query Processing Steps

5.3 SQL COMMANDS (S-17, 18)

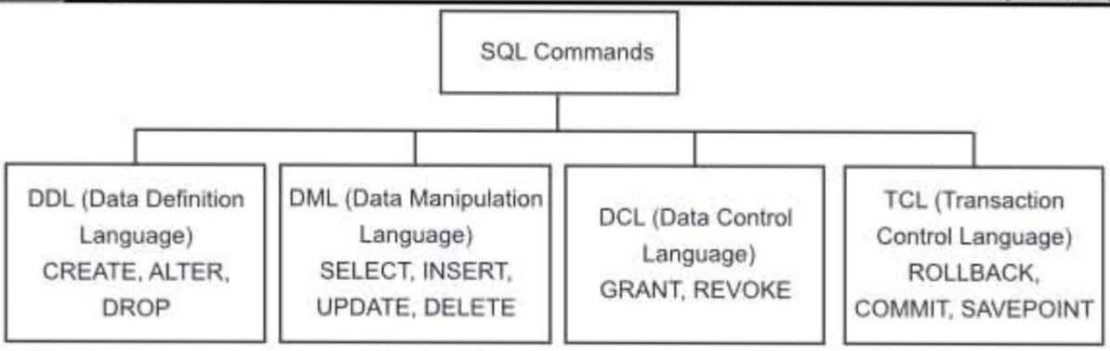


Fig. 5.3: SQL Commands

- As the name suggests DDL commands are used to define database structure or schema. DDL SQL commands are used for creating, modifying, and dropping the structure of database objects. Example : commands CREATE, ALTER, DROP etc.
- DML commands are used to change or alter data with the database or schema. DML SQL commands are used for storing, retrieving, modifying, and deleting data. Example : commands INSERT, UPDATE etc.
- DCL SQL commands are used for providing security to database objects. These statements are use to control access or privileges. Examples : GRANT, REVOKE etc.
- TCL commands control transactions. For example, COMMIT and ROLLBACK. COMMIT indicates successful termination while ROLLBACK indicates unsuccessful termination.

Data Types in SQL: **(W-17, S-18)**

(i) Boolean

- A Boolean data type can have one of three possible values: true, false or null. We use Boolean or bool reserved word to declare a column with Boolean data type. While inserting a value into a boolean column PostgreSQL converts it into appropriate boolean values.

(ii) Character

There are three character data types: CHAR, VARCHAR and TEXT

- **CHAR(n) :** It is fixed length character string with space padded. If we insert a string having length shorter than the length of the column, PostgreSQL pad spaces. Trying to insert a string having length greater than the length of the column raises an error message.
- **VARCHAR(n) :** It is variable length character string. PostgreSQL does not pad spaces when the length of the string to be stored is less than the length of the column.
- **TEXT :** It is the variable length character string with unlimited length.

(iii) Numeric

1. **Integers**
   - **Small Integer :** SMALLINT is a 2 byte signed integer. Its range is from -32768 to 32767.
   - **Integer :** INT is a 4 byte integer. Its range is from -2147483648 to 2147483647.
   - **Serial:** Similar to AUTO_INCREMENT column in MySQL. PostgreSQL automatically generate and populate values into the SERIAL column.
2. **Floating-point numbers**
   - **float(n) :** A floating point number with precision at least 'n', up to a maximum of 8 bytes.
   - **real or float8 :** 4 byte floating point number.
   - **numeric or numeric(m,n) :** A real number with 'm' digits with 'n' number after the decimal point.

(iv) Temporal

- **DATE :** It stores only dates.
- **TIME :** It stores the time of the day values.
- **TIMESTAMP:** It stores both date and time values.

(v) Arrays

- In PostgreSQL, we can store an array of strings, integers etc. in array columns.

(vi) JSON

- JSON stands for JavaScript Object Notation. It is a lightweight format for storing and transporting data. PostgreSQL has two JSON data types namely, JSON and JSONB for storing JSON data. JSON stores plain JSON data. JSONB stores JSON data in a binary format.

(vii) UUID

- The UUID allows you to store Universal Unique Identifiers. The UUID guarantees a better uniqueness than SERIAL. It is used to hide sensitive data such as values of id in URL.

(viii) Special data types

PostgreSQL provides data types related to geometric and network.

- box : a rectangular box
- point : a point on a plane(geometric pair of numbers)
- line : a set of points
- lseg: a finite line segment
- polygon : a polygon
- inet : an IPv4 and IPv6 address
- macaddr : MAC address

5.4 DDL COMMANDS WITH EXAMPLES (S-17)

- DDL refers to "DATA Definition Language". It deals with database schemas and descriptions.

1. CREATE TABLE command

We can create a table(relation) by using CREATE TABLE statement. The SQL relation is defined by using create table command by giving a table name and specifying attributes and constraints.

Syntax:

```
CREATE TABLE table_name (
   column_1   datatype,
   column_2   datatype,
      column_3  datatype,
      column_4  datatype,
   …
   column_N  datatype,
   PRIMARY KEY(one or more columns)
);
```

Integrity Constraints:

- Integrity Constraints are used to apply business rules for the database tables. The constraints available in SQL are Foreign Key, Not Null, Unique, Check. Constraints can be defined in two ways:
  (1) The constraints can be specified immediately after the column definition. This is called column-level definition.
  (2) The constraints can be specified after all the columns are defined. This is called table-level definition.

(i) Primary key:

This constraint defines a column or combination of columns which uniquely identifies each row in the table.

Define a Primary Key at Column Level:

Syntax:

```
column name datatype PRIMARY KEY
```

Define a Primary Key at Table Level:

Syntax:

```
[CONSTRAINT constraint_name] PRIMARY KEY
(column_name1,column_name2,..)
```

- column_name1, column_name2 are the names of the columns which define the primary key.

- The syntax within the bracket i.e. [CONSTRAINT constraint_name] is optional.

For Example: To create an employee table with Primary Key constraint, the query would be like.

Primary Key at Column Level:

```
CREATE TABLE employee
( id numeric(5) PRIMARY KEY,
name char(20),
dept char(10),
age numeric(2),
salary numeric(10),
location char(10));
```

Primary Key at Table Level:

```
CREATE TABLE employee
(id numeric(5),
name char(20),
dept char(10),
age numeric(2),
salary numeric(10),
location char(10),
CONSTRAINT emp_id_pk PRIMARY KEY(id)
);
```

Primary Key at Table Level:

```
CREATE TABLE employee
( id numeric(5) NOT NULL,
name char(20),
dept char(10),
age numeric(2),
salary numeric(10),
location char(10));
ALTER TABLE employee ADD CONSTRAINT PK_EMPLOYEE_ID PRIMARY KEY (id);
```

(ii) Foreign key or Referential Integrity:

This constraint identifies any column referencing the PRIMARY KEY in another table. It establishes a relationship between two columns in the same table or between different tables. For a column to be defined as a Foreign Key, it should be a defined as a Primary Key in the table which it is referring. One or more columns can be defined as Foreign key.

- **Syntax to define a Foreign key at Column Level:**

```
REFERENCES Referenced_Table_name(column_name)
```

- **Syntax to define a Foreign key at Table Level:**

```
[CONSTRAINT constraint_name] FOREIGN KEY(column_name) REFERENCES
referenced_table_name(column_name);
```

For example :

Foreign Key at column level:

```
CREATE TABLE dept( dept_no numeric(5) PRIMARY KEY, dname char(10));
CREATE TABLE emp( emp_id numeric(5) PRIMARY KEY, emp_name char(20),
                        emp_addr  char(50),dept_no  numeric(5)  REFERENCES
dept(dept_no));
```

Foreign Key at table level:

```
CREATE TABLE emp( emp_id numeric(5) PRIMARY KEY, emp_name char(20),
       emp_addr char(50),dept_no numeric(5),
       CONSTRAINT dpt_id_fk FOREIGN KEY(dep_no) REFERENCES dept(dept_no));
```

(iii)UNIQUE Key:

This constraint ensures that a column or a group of columns in each row have a distinct value. A column(s) can have a null value but the values cannot be duplicated.

- **Syntax to define a Unique key at Column Level:**

```
Column_name datatype [CONSTRAINT constraint_name] UNIQUE
```

- **Syntax to define a Unique key at Table Level:**

```
[CONSTRAINT constraint_name] UNIQUE(column_name)
```

For Example:

Unique key at Column Level:

```
CREATE TABLE student
( id numeric(5) PRIMARY KEY, name char(20),class char(10),age numeric(2),
location char(10) UNIQUE);
```

Unique key at Table Level:

```
CREATE TABLE student
( id numeric(5) PRIMARY KEY, name char(20), class char(10), age numeric(2),
location char(10), constraint uq_loc  unique(location));
```

OR

```
CREATE TABLE student
( id numeric(5) PRIMARY KEY, name char(20), class char(10), age numeric(2),
location char(10), unique(location));
```

(iv) NOT NULL Constraint:

This constraint ensures all rows in the table contain a definite value for the column which is specified as not null.

Syntax to define a NOT NULL constraint:

column_name datatype NOT NULL

For Example: To create a teacher table with not null columns, the query is as follows:

```
CREATE TABLE teacher
( t_id numeric(5) primary key,
t_name char(40) not null,
qual char(20) not null,
desg char(20) not null,
salary numeric(10) not null,
dname char(30) not null );
```

(v) Check Constraint:

This constraint defines a business rule on a column. All the rows must satisfy this rule. The constraint can be applied for a single column or a group of columns.

Syntax to define a Check constraint:

```
[CONSTRAINT constraint_name] CHECK (condition)
```

For Example: In the Student table to select the class of a student from the given set, i.e. ('fy','sy','ty'), the query would be like,

- **Check Constraint at Column Level:**

```
CREATE TABLE student(
roll_no numeric(5) primary key, name char(20) not null,
cls char(10) check(cls in('fy','sy','ty')), age numeric(2));
```

- **Check Constraint at Table Level:**

```
CREATE TABLE student
( roll_no numeric(5) primary key,
name char(20) not null,
cls char(10) not null,
CONSTRAINT cls_ck_student CHECK(cls in('fy','sy','ty'))
);
```

2. DROP TABLE command

If a table is not needed any more, the table is deleted. DROP TABLE command is used for this purpose. It removes the structure of the table as well as the data contained in it.

Syntax: `DROP TABLE tablename;`

For example: `DROP TABLE student;`

It will delete the student table from the database.

3. ALTER TABLE command **(S-18)**

Without deleting a relation, we may modify the existing relation. The modifications to the existing table can be performed by the ALTER TABLE command. The most important options are ADD, ALTER and DROP. With ALTER TABLE command you can add a column, drop a column, change the data type of a column, rename a column. ALTER

TABLE can also be used to set a default value for the column, to rename a table, to add a CHECK constraint to a column.

Options used with ALTER TABLE command:

- ADD : To add a new column to an existing table.
- DROP : To delete a column from a table.
- ALTER COLUMN column SET DATA TYPE datatype : To change the data type of a column.
- ALTER COLUMN column SET DEFAULT default_value : To set default value for a column of a table.

Variations of ALTER TABLE command:

- **To add a new column to an existing table**

  ```
  ALTER TABLE tablename ADD COLUMN column_name  DATATYPE;
  ```

  For example: to add a new column div to student table

  ```
  ALTER TABLE student ADD COLUMN div char(1) not null;
  ```

- **To delete a column from an existing table**

  ```
  ALTER TABLE tablename DROP COLUMN column_name;
  ```

  For example: to delete a column 'class'

  ```
  ALTER TABLE student DROP COLUMN class;
  ```

- **To rename a column to a new name**

  ```
  ALTER TABLE tablename RENAME COLUMN old_column_name TO new_column_name;
  ```

  For example: to rename a column 'cls' to 'class'

  ```
  ALTER TABLE student RENAME COLUMN cls TO class;
  ```

- **To change a default value of the column(to set or to drop)**

  ```
  ALTER TABLE tablename ALTER COLUMN column_name [SET DEFAULT value|DROP DEFAULT];
  ```

  For example: to set default value for column 'div' to 'A'

  ```
  ALTER TABLE student ALTER COLUMN div SET DEFAULT 'A';
  ```

  For example, to drop default value for column 'div' to 'A'

  ```
  ALTER TABLE student ALTER COLUMN div DROP DEFAULT;
  ```

- **To change the NOT NULL constraint (to set or to drop)**

  ```
  ALTER TABLE tablename ALTER COLUMN column_name [SET NOT NULL|DROP NOT NULL];
  ```

  For example: to set the not null constraint on 'div'

  ```
  ALTER TABLE student ALTER COLUMN div SET NOT NULL;
  ```

  For example: to drop the not null constraint on 'div'

  ```
  ALTER TABLE student ALTER COLUMN div DROP NOT NULL;
  ```

- **To change the data type of a column**

```
ALTER TABLE tablename ALTER COLUMN column_name SET DATA TYPE datatype;
```

For example: to change the data type of a column 'div' to varchar

```
ALTER TABLE student ALTER COLUMN div SET DATA TYPE varchar(2);
```

- **To rename a table**

```
ALTER TABLE tablename RENAME TO new_name;
```

For example: to rename 'student' table to 'stud'

```
ALTER TABLE student RENAME TO stud;
```

- **To add a constraint to a table**

```
ALTER TABLE tablename ADD CONSTRAINT constraint_name  definition;
```

For example: `ALTER TABLE stud ADD CONSTRAINT ck_div CHECK(div IN('fy','sy','ty'));`

5.5 BASIC STRUCTURE OF SQL QUERY (W-17, S-18)

For retrieving record from table, we use basic structure for data retrieval. The SQL expression has three clauses.

Syntax :

```
SELECT A1,A2,...,An
FROM r1,r2,r3,...,rm
WHERE p;
```

Where A_n represents attributes, r_m represents the tables and p is predicate(condition). Sometimes an asterisk(*) is used to select all attributes of a relation appearing in the FROM clause. The semicolon is the terminator of SQL expression.

The SELECT command has many clauses:

- **WHERE :** specifies the condition.
- **GROUP BY :** forms the groups according to a column so that an aggregate function can be applied to each individual group.
- **ORDER BY :** specifies the order in which to display the records(ascending or descending). The default order is ascending.
- **HAVING :** specifies condition on the groups formed by GROUP BY clause.
- **AS :** provides an alias for tables or columns.

Examples :

1. Display alphabetical list of students(according to the names) studying in 'A' division.

   SELECT roll_no,name FROM stud WHERE div='A' ORDER BY name;

2. Display divisionwise total number of students.

   SELECT div,count(*) FROM stud GROUP BY div;

3. Display the divisions whose strength is above 30.
   SELECT div FROM stud GROUP BY div HAVING count(*)>30;
4. Alias for table : Display the names of students studying in 'A' division.
   SELECT s.name FROM stud AS s WHERE div='A';
5. Alias for column : Display total number of students studying in 'A' division.
   SELECT count(*) AS total FROM stud WHERE div='A';

5.5.1 SQL Operators

- An operator is symbol or a sign. It is linked to some specific action. In SQL, we will look at different types of operators which are categorized according to their nature of work and behaviour.
- Types of Operators:

a. Arithmetic Operators:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
^	Exponentiation

b. Comparison Operators

Operator	Description
< (less than)	Returns TRUE when LEFT operand is less than the RIGHT operand
<= (less than or equal)	Returns TRUE when LEFT operand is less than or equal to the RIGHT operand
> (greater than)	Returns TRUE when LEFT operand is less than the RIGHT operand
>= (greater than or equal)	Returns TRUE when LEFT operand is less than or equal to the RIGHT operand
= (equal)	Returns TRUE when both the operands are equal
<>, != (not equal)	Returns TRUE when both the operands are not equal

For example, consider that student table has fields roll_no, name, class, div, age, address.

To display the names of the students whose age is at least 20, we write

```
SELECT name
FROM student
WHERE age >= 20;
```

c. **Logical Operators** : AND, OR, NOT

For example, consider that student table has fields roll_no, name, class, div, age, address.

To display the names of the students whose age is 20 and studying in F.Y.B.C.A., we write

```
SELECT name
FROM student
WHERE age = 20 AND class='F.Y.B.C.A.';
```

d. **Bitwise Operators**:

Operator	Description
&	Binary AND operator
|	Binary OR operator
~	Binary 1's complement
<<	Binary left shift operator
>>	Binary left shift operator
#	Bitwise XOR

e. **BETWEEN operator:** The BETWEEN operator retrieves values within a specified range. The lower_value and upper_value are included.

Syntax :

```
SELECT column_names
FROM table_name
WHERE column_name BETWEEN lower_value AND upper_value;
```

For example, consider that student table has fields roll_no, name, class, div, age, address.

To display the names of the students whose age is in between 20 to 25, we write

```
SELECT name
FROM student
WHERE age BETWEEN 20 AND 25;
```

5.5.2 Functions in PostgreSQL

- PostgreSQL includes a number of built-in functions that manipulate specific data types and return values.
- Let us see some of built-in functions.

a. Some string functions in PostgreSQL :

Function	Description
LOWER(string) OR LCASE(string)	Converts string to lowercase. e.g. select lower('HELLO') output : hello
UPPER(string) OR UCASE(string)	Converts string to uppercase. e.g. select upper('hello') output : HELLO
LENGTH(string)	Returns the number of characters in string.
LTRIM(string)	Removes leading whitespaces.
RTRIM(string)	Removes trailing whitespaces.
REVERSE(string)	Reverses the string.
LEFT(string,n)	Returns the left most 'n' number of characters in string. e.g. select left('hello',2) output : he
RIGHT(string,n)	Returns the right most 'n' number of characters in string. e.g. select right('hello',2) output : lo
SUBSTRING(string,start,length)	Returns a character string beginning from start upto length number of characters. The first position of string is 1. e.g. select substring('hello',1,2); output : he
CONCAT(string1,string2)	Returns concatenated string. e.g. select concat('good',' morning'); output : good morning

b. Some numeric functions in PostgreSQL :

Function	Description
ABS(expression)	Returns the positive value of expression. e.g. select abs(-11); output : 11

Contd...

CEILING(expression)	Returns the smallest integer >= expression. e.g. select ceiling(11.1*2); output : 23
FLOOR(expression)	Returns the largest integer <= expression. e.g. select floor(11.1*2); output : 22
ROUND(expression, n)	Returns expression rounded to n no. of places. e.g. select round(11.75,1); output : 11.8 select round(11.75,0); output : 12
POWER(num, n)	Returns a number 'num' raised to the power of 'n'. e.g. select power(2,3); output : 8
MOD(num, n)	Returns remainder after dividing num by n. e.g. select mod(11,2); output : 1

c. Some time and date functions in PostgreSQL :

Function	Description
current_date	Returns the current date in 'yyyy-mm-dd' format. e.g. select current_date; output : 2019-10-09
current_time	Returns the current time.
now( )	Returns the current local date and time.

5.6 SET OPERATIONS (S-17)

SQL provides some set operations like Union, Intersection and Except. The relations participating in the SQL operations must be compatible. i.e. they must have same set of attributes.

(a) UNION Operation: Union is one of the relational algebra operation which automatically eliminate duplicate values. It works just like OR operation.

Example: Find the student names who are either studying in 'A' or in 'B' division or both.

```
SELECT s.name FROM stud AS s WHERE s.div='A'
UNION
SELECT t.name FROM stud AS t WHERE t.div='B';
```

The Union operation automatically eliminates duplicates and the order is also maintained alphabetically.

If we want to keep the duplicate values in the result set, we should use a keyword UNION ALL.

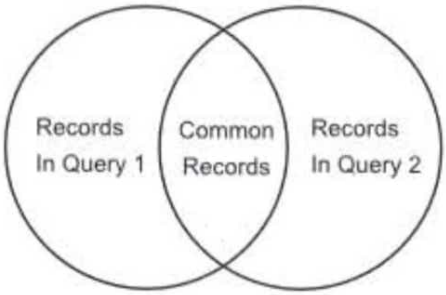


Fig. 5.4 Output of UNION operation

(b) INTERSECTION Operation: Intersection finds the common elements from both the relations. This operation automatically eliminates duplicate values. If we want to keep the duplicate in the result set, we should use a keyword INTERSECT ALL.

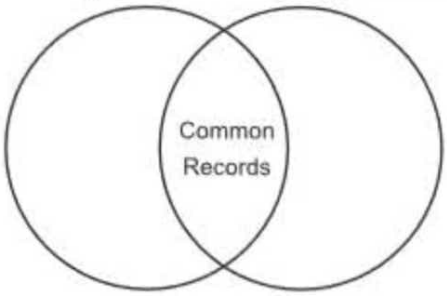


Fig. 5.5 Output of INTERSECT operation

Example: Find the name of the customers who are depositors of the bank and also have taken a loan from it.

```
SELECT name FROM depositor
INTERSECT
SELECT name FROM borrower;
```

(c) The EXCEPT operation: It is used to find the set difference between two result sets. Fig. 5.6 shows the Venn diagram about the output of EXCEPT operation.

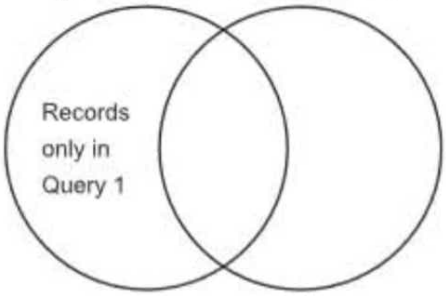


Fig. 5.6 Output of EXCEPT operation

Example: Find the name of the customers who are depositors of the bank and but have not taken any loan from it.

```
SELECT name FROM depositor
EXCEPT
SELECT name FROM borrower;
```

The EXCEPT operation automatically eliminates duplicate values. If we want the duplicate values, then use EXCEPT ALL.

5.6.1 Set Membership Operation (S-18)

There are two keywords used for set membership, i.e. IN and NOT IN.

The IN is used for checking the set membership and NOT IN is used for checking the absence of set membership.

Suppose there is a table student having fields rno, name, class, div.

Example : Find the name of the students who are studying in any of the 'FY', 'SY' , 'TY'classes.

```
SELECT name FROM student WHERE class IN('FY','SY','TY');
```

Example : Find the name of the students who are not studying in 'A' and 'B'divisions.

```
SELECT name FROM student WHERE div NOT IN('A','B');
```

String Operation (pattern matching) (W-17)

There are two pattern matching characters. LIKE operator is used for checking approximate equality.

(i) Underscore (_) : This character is used to match single character.

(ii) Percent (%) : This character is used to match any number of characters.

For example,

(i) Find the games whose name starts with 'vol' and ends with 'eyball' and between starting and ending strings there is one character.

```
SELECT * FROM game WHERE gname LIKE 'vol_eyball';
```

(ii) Find the games whose name starts ends with 'ball'.

```
SELECT * FROM game WHERE gname LIKE '%ball';
```

5.7 AGGREGATE FUNCTIONS (W-17, S-18)

SQL provides many built-in functions to perform operations on data. Aggregate functions operate on a group of values and returns a single value. SQL offers following built-in functions:

Example: Consider following emp table data for the aggregate functions.

Eid	name	age	salary
101	Anuradha	40	35000
102	Shirish	45	55000
103	Mahesh	37	50000
104	Ravindra	35	50000
105	Farukh	30	30000

1. **avg() :** The avg( ) function returns the average value of a numeric column.

Syntax :

```
Select avg(column_name) from table_name;
```

Example : Using SQL find average salary of emp.

```
Select avg(salary) from Emp;
```

Output:

avg
44000

2. **min() :** The min( ) function returns the smallest value of the selected column.

Syntax:

```
Select min(column_name) from table_name;
```

Example : Using SQL find minimum salary of emp.

```
Select min(salary) from emp;
```

Output:

min
30000

3. **max():** The max( ) function returns the largest value of the selected column.

Syntax:

```
Select max(column_name) from table_name;
```

Example: Using SQL find the highest salary of emp.

```
Select max(salary) from emp;
```

Output:

max
55000

4. **count() :** The count( ) function returns the number of rows that matches a specified criteria.

Syntax:

```
Select count(column_name) from table_name;
```

Example: Using SQL count the total number of employees.

```
Select count(name) from emp;
```

Output:

count
5

5. **sum()** : The sum( ) function returns the addition of the values within a column.

Syntax:

```
Select sum(column_name) from table_name;
```

Example: Using SQL count the total of the salaries paid to the employees.

```
Select sum(salary) from emp;
```

Output:

sum
220000

5.8 NULL VALUES

The SQL NULL is the term used to represent a missing or undefined value. A field with a NULL value is a field with no value. It is very important to understand that a NULL value is different than a zero value or a field that contains spaces.

```
Create table Player
( pl_id numeric(5) primary key,
pl_name varchar(30) not null,
addr varchar(40),
skill varchar(10));
```

In the above example, not null signifies that column should always accept a value for the given column. The columns 'addr' and 'skill' may have null values.

Consider the *Player* table having following records.

pl_id	pl_name	addr	skill
1	Ramesh	Main Road	Batsman
2	Rajesh	M G Road	Bowler
3	Jonny	East Road	Batsman
4	Vijay	East Road	
5	Yusuf		

IS NOT NULL example

```
Select pl_id,pl_name from Player where skill is not null;
```

Output :

pl_id	pl_name
1	Ramesh
2	Rajesh
3	Jonny

IS NULL example

```
Select pl_id,pl_name from Player where skill is null;
```

Output :

pl_id	pl_name
4	Vijay
5	Yusuf

5.9 NESTED SUB-QUERIES (W-17)

A query within a query is called nested query. It is referred also as nested sub-query. A sub-query is a SQL expression that is nested within another query. A sub-query is embedded within the WHERE clause. The results given by sub-query are used in the main(outer) query as a condition to restrict further the data to be fetched.

Sub-queries can be used with SELECT, INSERT, UPDATE and DELETE statements along with the operators like =, <, <=, >, >=, IN, BETWEEN etc.

- **Sub-queries in the SELECT statement:**

  The most frequently usage of sub-queries is in the SELECT statement.

  Syntax:

```
SELECT column_name [, column_name ]
FROM   table1 [, table2 ]
WHERE  column_name OPERATOR
   (SELECT column_name [, column_name ]
   FROM table1 [, table2 ]
   [WHERE])
```

  Example: Consider following emp table data for the nested sub-queries.

eid	name	age	salary
101	Anuradha	40	35000
102	Shirish	45	55000
103	Mahesh	37	50000
104	Ravindra	35	50000
105	Farukh	30	30000

Display the name of the highest paid employee.

```
SELECT name
FROM emp
WHERE salary=( SELECT max(salary)
               FROM emp );
```

Output:

name
Shirish

- **Sub-queries in the INSERT statement:**

The sub-queries can also be embedded in INSERT statement. The INSERT statement uses the data returned from the subquery to insert into another table.

Syntax:

```
INSERT INTO table_name [ (column1 [, column2 ]) ]
       SELECT [ *|column1 [, column2 ]
       FROM table1 [, table2 ]
       [WHERE VALUE OPERATOR ]
```

Example:

Consider that there is emp_backup table which is having the same structure as that of emp.

1. To copy all records from emp table into emp_backup we write:

```
INSERT INTO emp_backup SELECT * FROM emp;
```

2. To copy only specific records into emp_backup we write:

```
INSERT INTO emp_backup SELECT * FROM emp WHERE eid BETWEEN 101 AND 103;
```

In the second example, only three records will be copied into emp_backup table.

- **Sub-queries in the UPDATE statement:**

The sub-queries can also be used in the UPDATE statement. Either one or multiple columns in the table can be updated when using a subquery with the UPDATE statement.

Syntax:

```
UPDATE table
SET column_name = new_value
[ WHERE COLUMN_NAME OPERATOR
(SELECT COLUMN_NAME
FROM TABLE_NAME  [ WHERE  CONDITION]
    )
];
```

Example:

Consider that, we have emp_backup table for taking the backup of emp table.

The following example updates the salaries of those employees by 25% whose age is between 35 to 40.

```
UPDATE emp_backup
SET salary=salary*1.25
WHERE EID
IN (SELECT EID
FROM EMP
WHERE AGE BETWEEN 35 AND 40);
```

Output: emp_backup table contents after executing above query

eid	name	age	salary
101	Anuradha	40	43750
102	Shirish	45	55000
103	Mahesh	37	62500
104	Ravindra	35	62500
105	Farukh	30	30000

- **Sub-queries in the DELETE statement:**

The sub-query can also be used in conjunction with the DELETE statement like SELECT, INSERT, UPDATE.

Syntax:

```
DELETE FROM TABLE_NAME
      [ WHERE COLUMN_NAME OPERATOR
      (SELECT COLUMN_NAME
      FROM TABLE_NAME  [ WHERE CONDITION]
           )
      ];
```

Example: Consider that, we have emp_backup table for taking the backup of emp table.

The following example deletes the records of those employees whose age is in between 35 to 40.

```
DELETE FROM emp_backup
WHERE AGE IN ( SELECT AGE FROM EMP
               WHERE AGE BETWEEN 35 AND 40 );
```

Output: the contents of emp_backup table after executing above delete query

eid	name	age	salary
102	Shirish	45	55000
105	Farukh	30	30000

5.10 MODIFICATIONS OF DATABASE

- DML commands are used for manipulating data in database.
- DML commands are used for storing, retrieving, modifying, and deleting data. Various DML commands are explained below:

1. INSERT INTO Command:

The INSERT statement is used to add new rows (records) of data to a table in the database.

Syntax:

```
INSERT INTO TABLE_NAME
[(column1, column2, column3,...columnN)]
VALUES (value1, value2, value3,...valueN);
```

Here, column1, column2,...columnN are the names of the columns in the table into which you want to insert data.

- You can populate data into a table through select statement over another table provided another table has a set of fields, which are required to populate first table. Here, is the syntax:

```
INSERT INTO first_table_name [(column1, column2, ... columnN)]
SELECT column1, column2, ...columnN
FROM second_table_name
[WHERE condition];
```

- Following statements would create four records in EMP table:

```
1. INSERT INTO EMP (eid,name,age,salary)
   VALUES (101, 'Anuradha', 40, 35000);
2. INSERT INTO EMP (eid,name,age,salary)
   VALUES (102, 'Shirish', 45, 55000);
3. INSERT INTO EMP (eid,name,age,salary)
   VALUES (103, 'Mahesh', 37, 50000);
4. INSERT INTO EMP (eid,name,age,salary)
   VALUES (104, 'Ravindra', 35, 50000);
```

- You can create a record in EMP table using second syntax as follows:

```
INSERT INTO EMP  VALUES (105, 'Farukh', 30, 30000);
```

- All the above statements would produce the following records in EMP table:

eid	name	age	salary
101	Anuradha	40	35000
102	Shirish	45	55000
103	Mahesh	37	50000
104	Ravindra	35	50000
105	Farukh	30	30000

2. UPDATE Command:

- The UPDATE statement is used to modify the existing rows in a table. We can use WHERE clause with UPDATE query to update selected rows otherwise all the rows would be affected.

Syntax:

```
UPDATE table_name
SET column1 = value1, column2 = value2...., columnN = valueN
WHERE [condition];
```

Example: Consider the above EMP table having five records.

```
Update the salary of an employee by 10% whose eid is 104;
UPDATE emp SET salary=salary*1.1 WHERE eid=104;
```

Output: Contents of EMP table after the execution of update command.

eid	name	age	salary
101	Anuradha	40	35000
102	Shirish	45	55000
103	Mahesh	37	50000
104	Ravindra	35	55000
105	Farukh	30	30000

- If we want to modify salaries of all employees, then we do not have to use WHERE clause. The query is as follows:

```
UPDATE emp SET salary=55000;
```

Output: Contents of EMP table after executing UPDATE query command.

eid	name	age	salary
101	Anuradha	40	55000
102	Shirish	45	55000
103	Mahesh	37	55000
104	Ravindra	35	55000
105	Farukh	30	55000

3. DELETE Command:

- The DELETE command is used to delete rows(records) from a table. We can use WHERE clause to delete specific records. If WHERE clause is omitted, then all the records from the table are deleted.

  Syntax :

```
DELETE FROM table_name
[WHERE condition];
Example : Delete the record of an employee whose eid is 105.
DELETE FROM emp WHERE eid=105;
```

  Output: Contents of EMP table after executing DELETE query

eid	name	age	salary
101	Anuradha	40	55000
102	Shirish	45	55000
103	Mahesh	37	55000
104	Ravindra	35	55000

- If we want to delete all records of emp table, we write following DELETE query.

```
DELETE FROM emp;
```

- After executing this command, emp table would have no record.
- The difference between DROP and DELETE is that DELETE can delete all records from a table(without deleting the structure of a table) while DROP deletes all records as well as the structure of a table.

5.11 SQL MECHANISMS FOR JOINING RELATIONS (S-17, 18)

- In SQL using JOIN keyword we can join two tables. A JOIN is a means for combining columns(fields) from two tables by using common values. SQL joins are used to combine rows from two or more tables.

  Syntax for joining two tables is:

```
SELECT col1, col2, col3...
FROM table_name1, table_name2
WHERE table_name1.col2 = table_name2.col1;
```

  Consider the following two tables,

Table : Game

g_id	gname	no_of_pl
1	Hockey	11
2	Cricket	11
3	Basketball	5
4	Kabaddi	7
5	Volleyball	6

Table : Player

pl_id	plname	skill	g_id
11	Virendra	striker	1
12	Andrew	defender	1
21	Sachin	batsman	2
22	Rahul	batsman	2
23	Kapil	all rounder	2
31	Rajendra	defender	3

We can join these two tables in SQL statement as follows.

```
SELECT pl_id, plname, skill, gname FROM
game, player
WHERE game.g_id=player.g_id;
```

- The output will show the details of the players along with the games in which they are participating.

pl_id	plname	skill	gname
11	Virendra	striker	Hockey
12	Andrew	defender	Hockey
21	Sachin	batsman	Cricket
22	Rahul	batsman	Cricket
23	Kapil	all rounder	Cricket
31	Rajendra	defender	Basketball

- Several operators can be used to join tables(=, <> etc.). The most common operator is the '=' (equal)operator.
- There are following types of joins available in SQL:
  1. **INNER JOIN:** It returns records when there is a match from both tables.
  2. **LEFT JOIN:** It returns all records from the left table, even if there are no matches in the right table.
  3. **RIGHT JOIN:** It returns all records from the right table, even if there are no matches found in the left table.
  4. **FULL JOIN:** It returns records when there is a match found in one of the tables.
  5. **SELF JOIN:** It Is used to join a table to itself as if there are two tables, renaming at least one table in the SQL statement.
  6. **CROSS JOIN:** It returns the Cartesian Product(Cross Product) of the two or more joined tables.

1. INNER JOIN

- It is the most frequently used join in SQL. It is also known as equijoin. The query compares each record(row) of first table with each record(row) of second table to find all pairs of records(rows) that satisfy the join condition. When the join condition is satisfied, column values for each matched pair of records are combined into result and unmatched pairs are discarded.

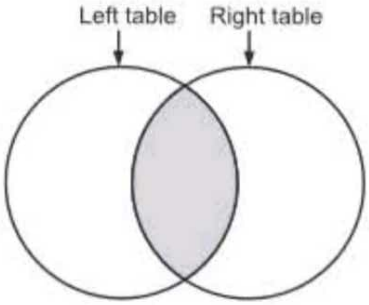


Fig. 5.7: Inner Join

INNER JOIN = All the matched pair of records from the both tables

Syntax:

```
SELECT table1.column1, table2.column2...
FROM table1
INNER JOIN table2
ON table1.common_field = table2.common_field;
```

Example: Consider the game and player tables which are shown in section 5.9

We join these two tables using INNER JOIN.

```
SELECT gname,no_of_pl,plname, skill
FROM game
INNER JOIN player
ON game.g_id=player.g_id;
```

Output :

gname	no_of_pl	plname	skill
Hockey	11	Virendra	striker
Hockey	11	Andrew	defender
Cricket	11	Sachin	batsman
Cricket	11	Rahul	batsman
Cricket	11	Kapil	all rounder
Basketball	5	Rajendra	defender

2. LEFT JOIN

- It returns all records from the left table, even if there are not matches found in the right table. It means that, if no matched records are found in the right table, the join will return a record in the result, with NULL in each column from right table. In some books, it is also called as LEFT OUTER JOIN.

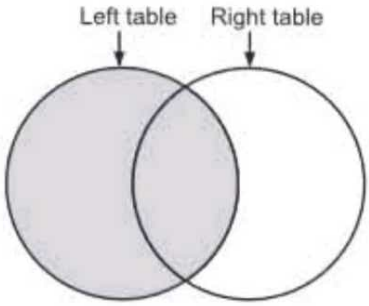


Fig. 5.8: Left Join

LEFT JOIN = All the values from the left table + matched values from the right table or NULL(for the unmatched records)

Syntax:

```
SELECT table1.column1, table2.column2...
FROM table1
LEFT JOIN table2
ON table1.common_field = table2.common_field;
```

Example: Consider the game and player tables which are shown in section 5.9

We join these two tables using LEFT JOIN.

```
SELECT gname,no_of_pl,plname, skill
FROM game
LEFT JOIN player
ON game.g_id=player.g_id;
```

Output:

gname	no_of_pl	plname	skill
Hockey	11	Virendra	striker
Hockey	11	Andrew	defender
Cricket	11	Sachin	batsman
Cricket	11	Rahul	batsman
Cricket	11	Kapil	all rounder
Basketball	5	Rajendra	defender
Volleyball	6		
Kabaddi	7		

- We observe that, for Volleyball and Kabaddi games there are no players. Hence, there is no value for Player Name and Skill in the columns.

3. RIGHT JOIN

- It returns all records from the right table, even if there are not matches found in the right table.
- It means that, if no matched records are found in the left table, the join will return a record in the result, with NULL in each column from left table. In some books, it is also called as RIGHT OUTER JOIN.

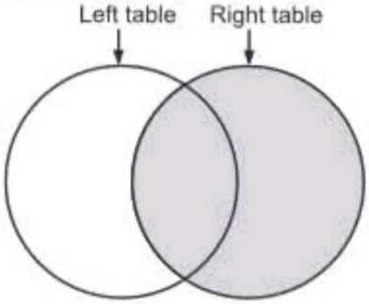


Fig. 5.9: Right Join

RIGHT JOIN = All the values from the right table + matched values from the left table or NULL(for the unmatched records)

Syntax:

```
SELECT table1.column1, table2.column2...
FROM table1
RIGHT JOIN table2
ON table1.common_field = table2.common_field;
```

Example: Consider the game and player tables which are shown in section 5.9

We join these two tables using RIGHT JOIN.

```
SELECT gname,no_of_pl,plname, skill
FROM game
RIGHT JOIN player
ON game.g_id=player.g_id;
```

Output:

gname	no_of_pl	plname	skill
Hockey	11	Virendra	striker
Hockey	11	Andrew	defender
Cricket	11	Sachin	batsman
Cricket	11	Rahul	batsman
Cricket	11	Kapil	all rounder
Basketball	5	Rajendra	defender

4. FULL JOIN

- The full join combines the results of both left and right outer joins.
- The full join is also known as full outer join.

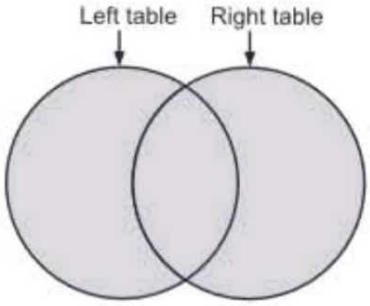


Fig. 5.10: Full Join

FULL JOIN = LEFT JOIN + RIGHT JOIN

Syntax :

```
SELECT table1.column1, table2.column2...
FROM table1
FULL JOIN table2
ON table1.common_field = table2.common_field;
```

Example: Consider the game and player tables which are shown in section 5.9.

We join these two tables using FULL JOIN.

```
SELECT gname,no_of_pl,plname, skill
FROM game
FULL JOIN player
ON game.g_id=player.g_id;
```

gname	no_of_pl	plname	skill
Hockey	11	Virendra	striker
Hockey	11	Andrew	defender
Cricket	11	Sachin	batsman
Cricket	11	Rahul	batsman
Cricket	11	Kapil	all rounder
Basketball	5	Rajendra	defender
Volleyball	6		
Kabaddi	7		

5. SELF JOIN

- This type of join is used to join a table with itself as if they are two separate tables. It renames at least one table in the SQL expression.

Syntax:

```
SELECT a.column_name, b.column_name...
FROM table1 a, table1 b
WHERE a.common_field = b.common_field;
```

Example: Consider the game and player tables which are shown in section 5.9

We join these two tables using SELF JOIN.

(i)
```
SELECT distinct m.g_id, m.gname, m.no_of_pl
FROM game m, game n
WHERE m.no_of_pl > n.no_of_pl;
```

OR

```
SELECT distinct game.g_id, game.gname, game.no_of_pl
FROM game, game n
WHERE game.no_of_pl > n.no_of_pl;
```

Output:

g_id	gname	no_of_pl
1	Hockey	11
2	Cricket	11
4	Kabaddi	7
5	Volleyball	6

(ii)
```
SELECT distinct m.g_id, m.gname, m.no_of_pl
FROM game m, game n
WHERE m.no_of_pl<n.no_of_pl;
```

OR

```
SELECT distinct game.g_id, game.gname, game.no_of_pl
FROM game, game n
WHERE game.no_of_pl<n.no_of_pl;
```

Output:

g_id	gname	no_of_pl
3	Basketball	5
4	Kabaddi	7
5	Volleyball	6

6. CROSS JOIN

- Cartesian Product(Cross Product) of the two or more joined tables. It means if in the left table has 5 records and right table has 6 records, then the result of cross join will contain 30 records. Cross join does not make any sense in practical situations.

Syntax:

```
SELECT table1.column1, table2.column2...
FROM  table1, table2 [, table3 ]
```

Example: Consider the game and player tables which are shown in section 5.9

We join these two tables using CROSS JOIN.

```
SELECT game.g_id, gname, pl_id, plname
FROM game, player;
```

Output :

g_id	gname	pl_id	plname
1	Hockey	11	Virendra
1	Hockey	12	Andrew
1	Hockey	21	Sachin
1	Hockey	22	Rahul
1	Hockey	23	Kapil
1	Hockey	31	Rajendra
2	Cricket	11	Virendra
2	Cricket	12	Andrew
2	Cricket	21	Sachin
2	Cricket	22	Rahul
2	Cricket	23	Kapil
2	Cricket	31	Rajendra
3	Basketball	11	Virendra
3	Basketball	12	Andrew
3	Basketball	21	Sachin
3	Basketball	22	Rahul
3	Basketball	23	Kapil
3	Basketball	31	Rajendra
4	Kabaddi	11	Virendra

Contd...

4	Kabaddi	12	Andrew
4	Kabaddi	21	Sachin
4	Kabaddi	22	Rahul
4	Kabaddi	23	Kapil
4	Kabaddi	31	Rajendra
5	Volleyball	11	Virendra
5	Volleyball	12	Andrew
5	Volleyball	21	Sachin
5	Volleyball	22	Rahul
5	Volleyball	23	Kapil
5	Volleyball	31	Rajendra

5.12 EXAMPLES OF SQL (CASES STUDIES) (S-17, 18; W-17)

1. Consider the following entities and relationships.

Owner (Licence_no, name, address, phone)

Car (car_no, model, colour)

Owner and Car are related with one-to-many relationships.

Create a RDB for the above and solve the following queries in SQL.

Solution : Owner and Car are the two relations given. Let's draw an E-R diagram with the given relationships. The E-R diagram is shown in Fig. 5.11.

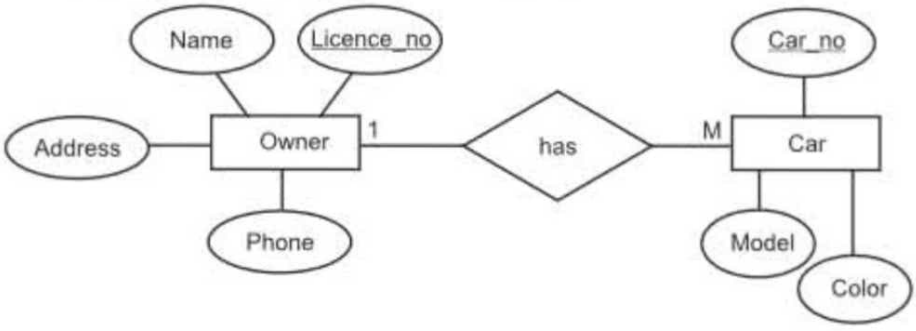


Fig. 5.11 : E-R diagram

After drawing E-R diagram, we have to convert it into the form of table with the given relationship.

The relationship is one-to-many, the primary key of relation one (called parent) is taken and added as a foreign key into the many (called child) relation. Let's draw on RDB which is shown in Fig. 5.12.

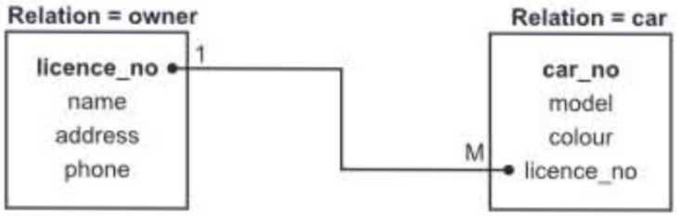


Fig. 5.12 : One-to-many relationship

Query 1: Find the name of owner of 'WagonR' and 'Alto' cars.

```
SELECT name, address, model
FROM car, owner
WHERE (model = 'WagonR' or model = 'Alto') and
owner.licence_no = car.licence_no;
```

Query 2: Insert a record in a car relation (for the owner whose license number = 1234).

```
INSERT INTO car
VALUES (201, 'Baleno', 'White', 1234);
```

Query 3: List all the models of owner 'Mr. Jeevan' having colour 'blue'.

```
SELECT model, colour
FROM car , owner
WHERE name = 'Mr. Jeevan' and colour = 'blue' and
owner.licence_no = car.licence_no;
```

2. Consider the following entities and relationships.

Machine (m_no, m_name, m_type, m_cost)

Part (p_no, p_name, p_desc)

Machine and Part are related with one-to-many relationships.

Create a RDB for the above and solve the following queries in SQL.

Solution : The one-to-many relationship is as shown in Fig. 5.13.

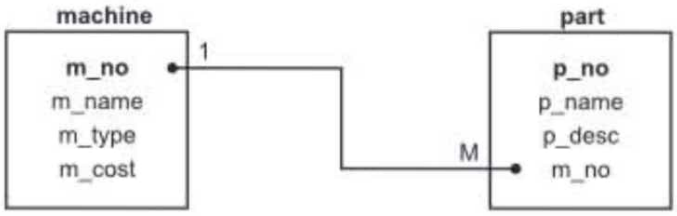


Fig. 5.13: One-to-many

Query 1: Increase the cost of machine by 5%.

```
UPDATE machine SET m_cost = m_cost + (m_cost * 0.05);
```

Query 2: List all the machines whose cost > 1,00,000.

```
SELECT m_name, m_cost
FROM machine
WHERE m_cost > 100000;
```

Query 3: List all the machines along with parts having description as "hardware".

```
SELECT m_name, p_name
FROM machine, part
WHERE machine.m_no = part.m_no and part.p_desc = 'hardware';
```

3. Consider the following entities and relationships.

Company (c_id, c_product, c_name, region, state)

Branches (b_product, city)

Company and Branches are related with one-to-many relationships.

Create a RDB for the above and solve the following queries in SQL.

Solution : The one-to-many relationship is as shown in Fig. 5.14.

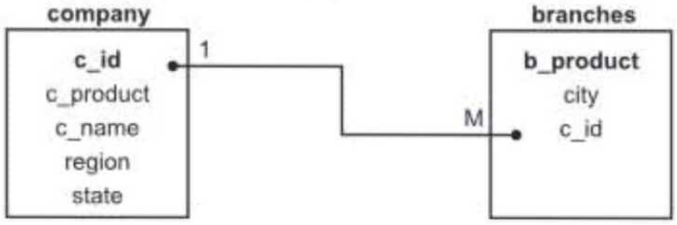


Fig. 5.14: One-to-many

Let's write queries:

Query 1: List all the cities having branch product 'washing machine' and 'microwave'

```
SELECT city, b_product
FROM branches, company
WHERE brances.c_id = company.c_id and (b_product = 'washing
   machine' or b_product='microwave');
```

Query 2: Display all the states whose branch product is 'music system'.

```
SELECT state
FROM branches, company
WHERE branches.c_id = company.c_id and b_product = 'music system';
```

Query 3: Delete all the branch details of city 'New York'.

```
DELETE
FROM branches
WHERE city='New York';
```

Query 4: Print city wise branches in sorted order.

```
SELECT city, c_name
FROM branches, company
WHERE branches.c_id = company.c_id
ORDER BY city;
```

4. Consider the following entities and relationships.

Teacher (t_no, t_name, college_name, dept)

E_Test(e_no, test_name)

Teacher and E_test are related with many-to-many relationships.

Create a RDB for the above and solve the following queries in SQL.

Solution : The relationship is many-to-many. So, we have to create a new table with the primary keys. The RDB is as shown in Fig. 5.15.

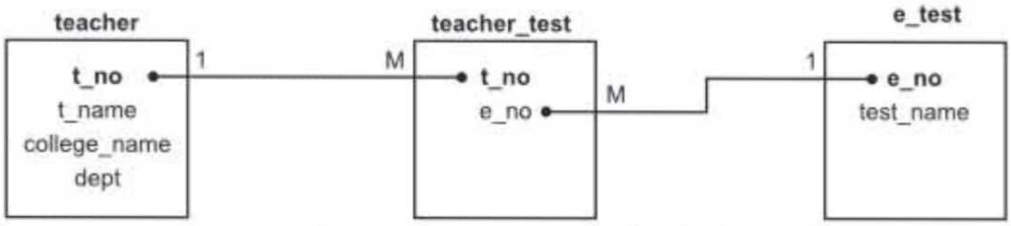


Fig. 5.15: Many-to-many relationship

Query 1 : Display the name of teachers who have either qualified "SET" or "NET".

```
SELECT t_name, college_name, test_name, dept
FROM teacher, teacher_test, e_test
WHERE teacher.t_no = teacher_test.t_no and
            e_test.e_no = teacher_test.e_no and (test_name = 'SET' or
                                                test_name = 'NET')
```

Query 2: List examinationwise list of teachers who have passed the respective examination

```
SELECT test_name, t_name, college_name, dept
FROM teacher, e_test, teacher_test
WHERE teacher.t_no = teacher_test.t_no and
e_test.e_no = teacher_test.e_no
GROUP BY test_name, t_name, college_name, dept;
```

Query 3 : Print the total number of teachers passing the respective exams.

```
SELECT test_name, count(*) as total
FROM teacher, e_test, teacher_test
WHERE teacher.t_no = teacher_test.t_no and
e_test.e_no=teacher_test.e_no
GROUP BY test_name;
```

5. Consider the following Entities and Relationships.

Country (con_code, name, capital)

Population (pop_code, popl)

Country and Population are related with One-to-One relationships.

Create a RDB for the above and solve the following queries in SQL.

Solution : The relationship given is one-to-one. There is no need to create a new table or take primary key and add it to other table.

The two tables will be as follows:

Population(pop_code, popl)

Country(con_code, name, capital, pop_code)

Pop_code in country acts as a foreign key with unique reference to its parent table.

Query 1: Give the name and population of country whose capital is 'Tokyo'.

```
SELECT name, popl
FROM Population p, Country c
WHERE p.pop_code=c.pop_code and capital = 'Tokyo';
```

Query 2: Give the name of all the countries whose population is greater than 50,00,000.

```
SELECT name, capital, popl
FROM Population, Country
WHERE Population.pop_code = Country.pop_code and popl > 5000000;
```

Query 3: Display the details of countries whose country name ends with 'ia' string.

```
SELECT *
FROM Country
WHERE name LIKE '%ia';
```

---

6. Consider the following entities and relationships

Wholesaler (w_no, w_name, addr, city)

Product (p_no, p_name)

Wholesaler and Product are related with many-to-many relationships.

Create a RDB for the above and solve the following queries in SQL.

Solution : Many-to-many relationship allows us to create new table. The RDB is as shown in Fig. 5.16.

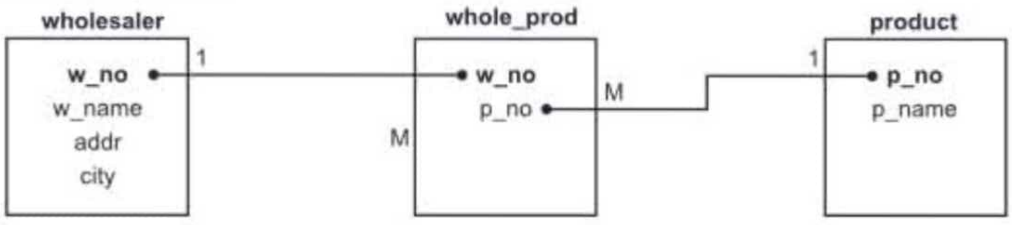


Fig. 5.16: Many-to-many relationship

Query 1: List all the wholesalers of product "books".

```
SELECT w_name, city
FROM wholesaler as w, product as p, Whole_prod as r
WHERE w.w_no = r.w_no and p.p_no = r.p_no and p.p_name = 'books';
```

Query 2: Count the no of wholesaler in the city "Ahmednagar".

```
SELECT COUNT (*)  as wholesaler_from_ahmednagar
FROM wholesaler
WHERE city = 'Ahmednagar';
```

Query 3 : Insert a record in wholesaler relation.

```
INSERT INTO wholesaler
VALUES (111, "Ajay Traders", "main road", "Ahmednagar");
```

Query 4: Print wholesalerwise product.

```
SELECT DISTINCT w_name, city, p_name
FROM wholesaler as w, product as p, whole_prod as r
WHERE w.w_no=r.w_no and p.p_no=r.p_no;
```

7. Consider the following entities and relationships.

Politician (p_no, p_name, p_desc, constituency)

Party(party_code, party_name)

Politician and party are related with many-to-one relationship. Create a RDB for the above and solve the following queries in SQL.

Solution: The relationship is many-to-one. The primary key of party is taken as foreign in politician table.

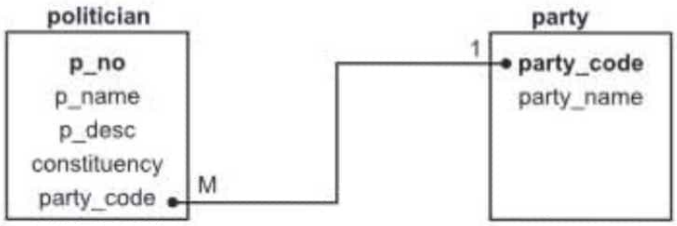


Fig. 5.17 Many-to-one relationship

Query 1 : Count the no. of politicians having political description as a 'member of parliament'.

```
SELECT count(p_name) as Total_MPs
FROM politician
WHERE p_desc = 'member of parliament';
```

Query 2 : Give the party name of politician 'Anil Kumar'.

```
SELECT party_name
FROM party, politician
WHERE politician.p_name = 'Anil Kumar' and
party.party_code = politician.party_code;
```

Query 3 : List party wise list of politicians of 'Ahmednagar' constituency.

```
SELECT politician.p_name, party.party_name, politician.party_name,
politician.constituency
FROM party, politician
WHERE politician.constituency = 'Ahmednagar' and
party.party_code=politician.party_code;
```

Query 4 : Find the total no. of politicians of each party for 'Pune' constituency.

```
SELECT party_name, count(p_name) as no_of_politician
FROM party, politician
WHERE party.party_code=politician.party_code and
politician.constituency = 'Pune'
GROUP BY party.party_name;
```

---

8. Consider the following entities and relationships.

Game(g_no, gname, no_of_player, coach_name, captain)

Player(p_no, p_name)

Game and player are related with many-to-many relationship. Create a RDB for the above and solve the following queries in SQL.

Solution : The table design for the above example is as follows:

```
Game(g_no, gname, no_of_player, coach_name, captain)
Player(p_no, p_name)
G_P(g_no,p_no)
```

Query 1: List the name of the players playing 'Volleyball'.

```
SELECT p_name
FROM game as g, player as p, G_P as r
WHERE p.p_no = r.p_no and g.g_no = r.g_no and g.gname='Volleyball';
```

Query 2: List the name of the players playing the game which ends with 'ball' string.

```
SELECT p_name, gname
FROM game as g, player as p, G_P as r
WHERE p.p_no = r.p_no and g.g_no = r.g_no and g.gname LIKE '%ball';
```

Query 3: Count the no. of players who are guided by 'Anderson'.

```
SELECT count(*) as no_of_players
FROM game as g, player as p, G_P as r
WHERE p.p_no = r.p_no and g.g_no = r.g_no and g.coach_name = 'Anderson';
```

Query 4: Display the games which require at least 5 players.

```
SELECT gname
FROM game
WHERE no_of_player >= 5;
```

9. Consider the following entities and relationships.

Sailors(sid, sname, rating, age)

Boats(bid, bname, colour)

Reserves(sid, bid, day)

Solve the following queries in SQL.

Solution : There is many-to-many relationship between Sailors and Boats.

Query 1: Find the details of sailors whose rating is at least 9.

```
SELECT *
FROM Sailors
WHERE rating >= 9;
```

Query 2: Find the sailor id of sailors who have reserved a 'Red' boat.

```
SELECT R.sid
FROM Boats as B, Reserves as R
WHERE B.bid = R.bid and B.colour = 'Red';
```

Query 3: Find the colours of boats reserved by 'Ravi'.

```
SELECT B.colour
FROM Boats as B, Reserves as R, Sailors as S
WHERE B.bid = R.bid and S.sid=R.sid and S.sname = 'Ravi';
```

Query 4: Find the names of sailors who have reserved at least one boat.

```
SELECT S.sname
FROM Reserves as R, Sailors as S
WHERE S.sid = R.sid;
```

Query 5: Find the name and ages of sailors whose name start and ends with P.

```
SELECT sname
FROM Sailors
WHERE sname LIKE 'P%P';
```

Query 6: Find the name of the boats which have been reserved by more than one sailor.

```
SELECT Boats.bname
FROM Boats
WHERE Boats.bid IN
(SELECT Boats.bid FROM Reserves
GROUP BY Reserves.bid
HAVING count(*) > 1);
```

Summary

- SQL is accepted standard for database. It is used to create, maintain and retrieve the relational database.
- DDL (Data Definition Language) is used to define the structure of the database tables.
- DML (Data Manipulation Language) commands are used to retrieve, insert, update or delete information from database tables.
- DCL (Data Control Language) consists of commands to set the permissions and control the database system.
- TCL (Transaction Control Language) commands are related with the transaction.
- Nested queries are nothing but sub queries that provide the data to the enclosing query.
- Aggregate functions work on a group of records and returns single value.
- For pattern matching LIKE operator is used.
- There are different types of joins: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, SELF JOIN, CROSS JOIN. The most commonly used join is INNER JOIN.

Check Your Understanding

Q.1. Multiple Choice Questions:

1. What is the full form of SQL?
   (a) Standard Query Language (b) Structured Query Language
   (c) Simple Query Language (d) none of these
2. Which of the following is not set membership operator?
   (a) INTERSECT (b) IN
   (c) NOT IN (d) all of these
3. Which of the following is not a set operation in SQL?
   (a) INTERSECT (b) UNION
   (c) CREATE (d) none of these

4. Which of the following is not an aggregate function in SQL?
   (a) avg (b) max
   (c) min (d) val
5. Which of the following is not a DDL command?
   (a) CREATE (b) ALTER
   (c) DROP (d) DELETE
6. Which of the following is a DML command?
   (a) CREATE (b) ALTER
   (c) DROP (d) DELETE
7. FULL JOIN = ... + RIGHT JOIN
   (a) LEFT JOIN (b) CROSS JOIN
   (c) SELF JOIN (d) all of these
8. In SQL, the count function returns the number of .....
   (a) columns (b) tables
   (c) values (d) groups
9. Which of the following symbol matches single character in pattern matching?
   (a) _ (b) %
   (c) $ (d) @

Ans.: (1) (b) (2) (a) (3) c (4) (d)
(5) d) (6) (d) (7) a (8) (c)
(9) (a)

Q.2. State True/False.

1. The embedded form of SQL is designed which is used in general purpose programming languages.
2. DDL command modifies the structure of a table.
3. DELETE command deletes the structure of a table.
4. A table can have any number of primary keys.
5. NOT NULL definition for a column ensures that the column must have some value for all rows.
6. SET DATA TYPE clause in ALTER TABLE is used to change the data type of a column.
7. GRANT is a DDL command.
8. ROLLBACK is a DCL command.
9. TIMESTAMP stores both date and time values.

Ans.: (1) True (2) True (3) False (4) False
(5) True (6) True (7) False (8) False
(9) True

Practice Questions

Q.1 Answer the following questions in brief.

1. What is SQL?
2. What are the aggregate functions provided in SQL?
3. Why CREATE TABLE is a DDL command?
4. Explain in brief UNION operation.
5. What are set membership operators in SQL?
6. State the difference between DROP and DELETE command.
7. State the difference between ALTER and UPDATE command.
8. What is a nested subquery?
9. State any two clauses used with ALTER TABLE command.

Q.2 Answer the following questions.

1. How database tables are created and maintained using SQL?
2. Explain the different clauses associated with SELECT command in SQL.
3. Explain the aggregate functions used in SQL.
4. Explain the different data types in PostgreSQL.
5. Explain about the integrity constraints.
6. Explain in detail about set operations in SQL.
7. Describe the different types of JOIN.
8. Explain the different DML commands.
8. Write about advantages of SQL.

Q.3. Define the terms.

(a) Unique Key
(b) Primary Key
(c) Foreign Key
(d) Except Operation
(e) Intersect Operation
(f) Union Operation
(g) In Operation
(h) Cross Join
(i) Self Join

Previous Exams Questions

Summer 2017

1. Inner join is also known as ..... . **[1 M]**

(i) Full join
(ii) Equi join
(iii) Natural join
(iv) Semijoin

Ans.: Refer to Section 5.9.

2. State various comonds that come under DDL.

Ans.: Refer to Section 5.2.

3. Explain various set operations with syntax and example.

Ans.: Refer to Section 5.4.

4. Consider the following relations,

   Machine (m_no, m_name, m_type, m_cost)

   Part (p_no, p_name, p_desc).

   Machine and part are related with one to may relationship.

   Create RDB and solve queries

   (i) Increase the cost of machine by 10%.

   (ii) Delete all machines having particulars 'wheel".

   (iii) List all machines whose cost > 1,00,000.

Ans.: Refer to Section 5.10.

5. Write a show note on components of SQL.

Ans.: Refer to Section 5.2.

6. What is Join? What is basic condition for taking join of two relations? Explain any two types of join with example.

Ans.: Refer to Section 5.9.

7. Consider the following relations:

   Docotr (docno, name, specialization)

   Hospital (hospno, ame, addr.)

   Doctor and Hospital are related with many relationship, with attribute day, create RDB for above and solve the queries.

   (i) List names of doctors visiting 'Nobel Hospital on Monday'.

   (ii) Delete all doctors with specialization 'gynaec'.

Ans.: Refer to Section 5.10.

Winter 2017

1. Which of the following is no an aggregate function? **1 M**

   (i) min (ii) max

   (iii) avg (iv) order by

Ans.: Refer to Section 5.5.

2. Give an example of nested subquery.

Ans.: Refer to Section 5.7.

3. Consider the following relations:

   Musician (m_no, m_name, age, m_city)

   Instrument (i_no, i_name)

Play (m_no, i_no)
Solve the following queries:
(i) List all the musicians having age between 30 and 40 years.
(ii) List all violin' players who live in 'Mumbai' and their age is below 30.

4. Explain 'Group by' and 'Order by' clauses in SQL with example.

Ans.: Refer to Section 5.3.

5. Explain any five aggregate functions in detail.

Ans.: Refer to Section 5.5.

6. What are different data types available in SQL? Explain in detail.

Ans.: Refer to Section 5.2.

7. Consider the following relations and solve the queries:
Item (i_code, i_name, price)
Order (o_code, date, cust_name)
Item_order (i_code, o_code, qty)
(i) List all order numbers along with different items.
(ii) List all order before 4th October, 2010.
(iii) List all items along with their price.

Ans.: Refer to Section 5.10.

8. Explain pattern matching operators in SQL.

Ans.: Refer to Section 5.4.1.

Summer 2018

1. SQL uses logical connectives not as ..... . **1 M**
(i) and (ii) or
(iii) not (iv) as

Ans.: Refer to Section 5.3.

2. Explain the use of order by clause.

Ans.: Refer to Section 5.3.

3. Explain basic structure at SQL queries.

Ans.: Refer to Section 5.3.

4. What are the types of SQL joins?

Ans.: Refer to Section 5.9.

5. What are SQL clauses to modify to database? Explain.

Ans.: Refer to Section 5.2.

6. Explain set membership and set comparison in SQL.

Ans.: Refer to Section 5.4.1.

7. Explain different data types in SQL.

Ans.: Refer to Section 5.2.

❖❖❖

6... Relational Database Design

Learning Objectives...

- To know about the basics of relational database design.
- To understand the pitfalls in relational database design.
- To learn the importance of normalization in database design and various normal forms.

6.1 INTRODUCTION

- The goal of relational database design is to design tables that store information without unnecessary duplication of the data. It also helps to retrieve the information easily. One approach of database design is to follow the process of normalization. Normalization is a systematic process of organizing data efficiently in a database. There are different normal forms. A good database design ensures that the data that we want to store is consistent.

6.2 PITFALLS IN RELATIONAL DATABASE DESIGN (S-17, W-17, S-18)

- In order to understand the pitfalls in relational database design, we consider a simple example of GAME_PLAYER relation.

Table 6.1 : GAME_PLAYER

Player_No	Name	City	Game_No	Gname
11	Smita	Pune	1	Basketball
11	Smita	Pune	2	Volleyball
12	Shirish	Pune	3	Cricket
13	Sanjiv	Mumbai	1	Basketball

Contd...

13	Sanjiv	Mumbai	3	Cricket
14	Mandar	Baramati	3	Cricket
14	Mandar	Baramati	4	Skating
15	Kaustubh	Solapur	3	Cricket
16	Mahesh	Ahmednagar	3	Cricket

- By observing above table GAME_PLAYER, some pitfalls are listed below:

1. **Redundancy:**
   - We can observe that, player information is repeated for each game the player is participating in. Also, game information is repeated for each participating player. This is called as redundancy. This results in requirement of more storage space and resources.
2. **Problems related with insertion of records:**
   - To insert a new player record into GAME_PLAYER, we have to add the game information along with the player information. If the player has not started to play a game, then we have to add NULL values in the game information.
   - It creates problem when we have to insert information of a new game which is not played by any player as yet. We can add such game information by placing NULL values in the player related columns (Player_No, Name, City).
3. **Problems related with deletion of records:**
   - If we delete a player record from GAME_PLAYER table and suppose that the player to be deleted is the only player playing that game, then the game information is lost from the database.
   - For example, consider a row from GAME_PLAYER table

Table 6.2

Player_No	Name	City	Game_No	Gname
14	Mandar	Baramati	4	Skating

   - If we delete the above row, then information related with 'Skating' game is lost.
4. **Problems related with modification of records:**
   - If we want to modify the city of a player in GAME_PLAYER table, then we have to modify all the GAME_PLAYER records which shows the participation of that player.
   - For example, consider a row from GAME_PLAYER table. We want to update the city of player 'Sanjiv' to Solapur. We have two records in GAME_PLAYER of this player.

Table 6.3

Player_No	Name	City	Game_No	Gname
13	Sanjiv	Mumbai	1	Basketball
13	Sanjiv	Mumbai	3	Cricket

- If we only change the city of the first record to 'Solapur', then it will lead to inconsistency situation.

Table 6.4

Player_No	Name	City	Game_No	Gname
13	Sanjiv	Solapur	1	Basketball
13	Sanjiv	Mumbai	3	Cricket

- In the above table, the same player is showing two different cities which is unacceptable.

5. We may add many columns which are inapplicable for most of the records in the relation. Here we have to insert many NULL values in the inapplicable columns for the individual record. This results in wastage of storage space.

6.3 FUNCTIONAL DEPENDENCIES (S-17, 18, W-17)

- The important concept in relational database design is functional dependency. Functional dependency is represented by an arrow sign (→).
- For example, In a relation R with attributes X and Y, a functional dependency between X and Y is shown as $X \rightarrow Y$, we read this as 'X functionally determines Y'. Here, X (left hand side) is a determinant and Y (right hand side) is a dependent attribute.

6.3.1 Basic Concepts

- A functional dependency establishes a set of constraints among attributes. Given attributes X and Y (each of which may be set of attributes). Y is said to be functionally dependent on X, if a given value for each attribute X uniquely determines the value of the attributes in Y.
- **Definition :** Let R be any relation. X and Y are subsets (of attributes) of R. The functional dependency $X \rightarrow Y$ holds on R, if in any legal relation r(R), for all pairs of tuples t1 and t2 in r(R) such that t1[X] = t2[X] then t1[Y] = t2[Y] also holds.
- The above definition means, in order to hold the Functional Dependency $X \rightarrow Y$, if two tuples agree on the determinant attribute(s), then they must also agree on dependent attribute(s).
- **Example :** Consider the following table STUDENT_INFO.

Table 6.5 : STUDENT_INFO

Stud_id	Semester	Subject	Instructor
11	3	Quantitative Techniques	Rajesh
12	4	Quantitative Techniques	Brijesh
11	3	C Programming	Rahul
13	4	Quantitative Techniques	Shivam
13	4	Discrete Mathematics	Prashant

- By observing the above table, we note that whenever a two rows has same Stud_id, they also have the same Semester values. This shows that there is a functional dependency.

Stud_id → Semester

- But if adding a new row having a different value of semester would violate the definition of functional dependency and the Functional Dependency would not exist.
- Other Functional Dependencies are {Stud_id, Subject} → Instructor, {Stud_id, Subject} → {Instructor, Semester}
  - **Full functional dependency :** A functional dependency M → N is a full functional dependency if removing any attribute 'A' from set M results in violation of Functional Dependency M→N. In other words, for any attribute A∈M, (M − {A}) does not functionally determine N.
  - **Partial dependency :** A functional dependency M → N is a partial dependency, if after removing some attribute A from M the dependency is still valid. In other words, A∈M, (M − {A}) → N.

    For example, consider a relation PROJ_INFO (Emp_No, Proj_No, Duration, Emp_Name, Proj_Name, Proj_City) {Emp_No, Proj_No}→Duration is a full functional dependency. Removing any attribute from the determinant part violates the functional dependency.

    But the dependency, {Emp_No, Proj_No} → Emp_Name is partial dependency because Emp_No → Emp_Name holds.
  - **Trivial Dependency :** If the dependent is a subset of the determinant, trivial dependency exists. As per the name, trivial dependency is having little value.

    For example, MN → N is a trivial dependency.
  - **Non-trivial dependency :** In a non-trivial dependency, the dependent(right hand side) part is not a subset of the determinant (left hand side) part. In other words, M→N holds, where N is not a subset of M.
  - **Transitive Dependency :** It is one type of functional dependency in which a non prime attribute is determined by another non prime attribute. Suppose N and P

are the non prime attributes and M is the prime attribute of relation R. If $M \rightarrow N$, $N \rightarrow P$ then the transitive functional dependency between M and P is shown as $M \rightarrow P$ (M transitively determines P).

- **Multivalued Dependency :** This type of dependency arises when we have to repeat every value of one of the attributes of the relation with every value of other attributes to keep the relation instances consistent.
  Suppose R is a relation having M, N and P as three different attributes or sets of attributes. In the relation R, if for every value of M there exists a set of values for N and P, but the set of values for N and P are not dependent on each other, then N and P are multivalued dependent on M.
  That is $M \rightarrow\rightarrow N$ and a relation R a multivalued dependency of the form $M \rightarrow\rightarrow N$ is said to be trivial multi-valued dependency, if N is a subset of M or $M \cup N = R$.
  For example, Consider relation Car_Info as shown in Fig. 6.6. These multivalued dependencies can be indicated as:
  $A \rightarrow\rightarrow B$ is read as B is multi-dependent on A.

Table 6.6 : Car_Info

Model	Year	Colour
W2016	2016	White
W2016	2016	Silver
W2019	2019	Blue
W2019	2019	White
A2017	2017	White
A2018	2018	Red

- In the above example, Year and Colour are independent of each other and they are dependent on Model. Here these two columns are said to be multivalued dependent on Model of car.
  (i) Model $\rightarrow\rightarrow$Year
  (ii) Model$\rightarrow\rightarrow$Colour

6.3.2 Closure of set of Functional Dependencies (W-17)

Inference Rules (Armstrong's axioms)

- These rules were developed by William W. Armstrong in 1974. He developed a systematic approach to derive all functional dependencies that can be inferred from F. The technique to compute closure of functional dependencies F^+ is based on these axioms.

- Let R be a relation and M, N and P are sets of attributes in a relation R.
  (i) Reflexivity : If N is a subset of M, then M→N.
  (ii) Augmentation : If M →N, then MP→NP.
  (iii) Transitivity : If M→N and N→P then M→P.
  (iv) Union : If M→N and M→P, then M→NP.
  (v) Decomposition : If M→NP, then M→N and M→P.
  (vi) Pseudotransitivity : If M→N and QN→T, then QM→T.

- **For Example :** Consider the relation R = (A, B, C, D, E) and set of FDs defined on R, F as {A → B, CD → E, A → C, B → D, E → A}.
  Using the above inference rules, we compute F^+ as follows :
  (i) A → D : Using transitivity rule (A → B, B→ D ⇒ A → D).
  (ii) A → E : Using pseudo transitivity rule (A → C, CD → E ⇒ A → E)
  (iii) CD → A : Using transitivity rule (CD→ E, E→ A ⇒ CD→ A)
  (iv) BC → E : Using pseudo transitivity rule (B →D, CD→E ⇒ BC→ E)
  (v) A → BC : Using union rule (A→B, A→C ⇒ A→BC).

6.3.3 Closure of an Attribute Set

- Closure of a set of attributes X with respect to F is the set X^+ of all attributes that are functionally determined by X using F^+.
- The algorithm to find closure of an attribute set is as follows:

Step 1: Start

Step 2 : Initialize attribute closure to the attribute itself. i.e. $A^+ := A$;

Step 3. While there are changes to A^+ (i.e. the attribute closure) do

Step 3.1: For each functional dependency M→N ∈ F do

Step 3.1.1: if $A^+ \supseteq M$ then $A^+ := A^+ \cup N$;

Step 4. Stop

For Example : Consider F = {A→B, B→C, C→D}. Find closure of A, B, C and D.

Solution :

(i) $A^+ := A$
$A^+ := (A,B)$ in A→B, $A^+ \supseteq A$
$A^+ := (A,B,C)$ in B→C, $A^+ \supseteq B$
$A^+ := (A,B,C,D)$ in C→D, $A^+ \supseteq C$

(ii) $A^+ := B$
$A^+ := (B)$ in A→B, $A^+ \supseteq A$
$A^+ := (B,C)$ in B→C, $A^+ \supseteq B$
$A^+ := (B,C,D)$ in C→D, $A^+ \supseteq C$

(iii) $A^+ := C$
$A^+ := (C)$ in A→B, $A^+ \supseteq A$

$A^+ := (C)$ in $B \rightarrow C$, $A^+ \supseteq B$

$A^+ := (C,D)$ in $C \rightarrow D$, $A^+ \supseteq C$

(iv) $A^+ := D$

$A^+ := (D)$ in $A \rightarrow B$, $A^+ \supseteq A$

$A^+ := (D)$ in $B \rightarrow C$, $A^+ \supseteq B$

$A^+ := (D)$ in $C \rightarrow D$, $A^+ \supseteq C$

6.4 CONCEPT OF A SUPER KEY AND A PRIMARY KEY

- **Superkey :** A superkey of a relation schema R is a set of attributes S (which is a subset of R) that satisfy the condition that no two rows r_1 and r_2 in a relation state r of R will have $r_1[S]=r_2[S]$.
- The removal of any attribute from a superkey will cause superkey not to be a superkey anymore.
- Superkey is a set of attributes that uniquely identifies each record in a table. It is a superset of candidate keys. The difference between a superkey and a primary key is that a primary key is a minimal superkey.
- If closure of A (i.e. A^+) determines all the attributes of R, then A is called superkey.

Example : Consider a schema R(A,B,C,D,E) and F={AB→C, C→D, B→E}. Find superkey.

Solution: Find closure set of (A, B, C, AB). i.e the closure of all determinants in F.

$A^+ = \{A\}$

$B^+ = \{B, E\}$

$C^+ = \{C, D\}$

$(AB)^+ = \{A, B, C, D, E\}$

We observe that AB^+ contains all attributes of original relation R. Therefore, AB is the superkey.

Example : Consider a schema R(A,B,C,D,E) and F={AB→C, C→D, B→EA}. Find superkey.

Solution: Find closure set of (A, B, C, AB). i.e the closure of all determinants in F.

$A^+ = \{A\}$

$B^+ = \{B, E, A, C, D\}$

$C^+ = \{C, D\}$

$(AB)^+ = \{A, B, C, D, E\}$

Closures of both B and AB determines all attributes of original relation R. Therefore, B and AB are superkeys. Only B is the candidate key as it contains minimal attributes.

Example : Consider a schema R(A,B,C,D,E,H) and F={A→B, BC→D, E→C, D→A}. Which of the following are candidate keys of R?

AE, AEH, BEH, DEH

Solution: We compute closure of the attribute set.

$(AE)^+ = \{A, B, E, C, D\}$

$(AEH)^+ = \{A, E, H, B, C, D\}$
$(BEH)^+ = \{B, E, H, C, D, A\}$
$(DEH)^+ = \{D, E, H, C, A, B\}$

By observing the closures, only AE is not a candidate key. AEH, BEH and DEH are candidate keys of R.

6.5 CONCEPT OF DECOMPOSITION (S-17, W-18)

- It is required to eliminate redundancy from the schema. Decomposition means breaking up a relation schema into smaller relation schemas. In order to have a good database design it is necessary to decompose big relations into smaller ones. If a relational schema R has redundancy in the data, then decompose R into R_1 and R_2 schema. There are two properties, which should be maintained when we perform decomposition, it should be lossless join and dependency preserving.

Goals of Decomposition:

1. To eliminate redundancy by decomposing a relation into multiple relations.
2. It is also important to check that decomposition of relation does not lead to a bad design.

- In decomposition, a relation is replaced with a collection of smaller relations with specific relationship between them.
- Formally, the decomposition of a relation schema R is defined as its replacement by a set of relation schemas such that each relation schema contains a subset of the attributes of R.

Definition of Decomposition:

- Let R be a relation schema then a set of relation schemas $\{R_1, R_2, ..., R_n\}$ is a decomposition of R if,
  1. $R = R_1 \cup R_2 \cup .... \cup R_n$
  2. Each R_1 is a subset of R (for i = 1, 2, ..., n)
- This means that relation R contains attributes $A_1 ... A_n$. A decomposition of R consists of replacing R by two or more relations such that; Each new relation scheme contains a subset of the attributes of R (and no attributes that do not appear in R), and every attribute of R appears as an attribute of one of the new relations.

  For example : A relation R(x, y, z) this relation can be decomposed into 2 subsets i.e., $R_1(x, z)$ and $R_2(y, z)$. If we union R_1 and R_2, we get $R = R_1 \cup R_2$.

OR

- The decomposition of a relation scheme $R = \{A_1, A_2, A_3, ..., A_n\}$ is its replacement by a set of relation schemas $\{R_1, R_2, R_3, ..., R_m\}$ such that,

$$R = R_1 \bowtie R_2 \bowtie R_3 \bowtie ..... \bowtie R_m.$$

6.6 DESIRABLE PROPERTIES OF DECOMPOSITION (W-17)

i. Loss-Less Join Decomposition

Definition :

- Let R be relation schema and let F be a set of functional dependencies on R. Let R_1 and R_2 form a decomposition of R. This decomposition is a loss-less join decomposition of R if at least one of the following functional dependencies are in F^+.
  - i. $R_1 \cap R_2 \rightarrow R_1$
  - ii. $R_1 \cap R_2 \rightarrow R_2$

Algorithm for finding whether decomposition is lossless or not:

Step 1 : First check whether the union of all decomposed sub-relation should be equal to relation R.

$$R_1 \cup R_2 \cup R_3 \cup ... \cup R_n = R$$

Step 2 : Any two sub-relations Ri and Rj can be merged into Rij with $R_1 \cup R_2$ only if

i. $R_i \cap R_j \neq \varnothing$

ii. $R_i \cap R_j = A$, then closure of A i.e. $A^+ \supseteq R_i$

or

$R_i \cap R_j = A$, then closure of A i.e. $A^+ \supseteq R_j$

Step 3 : Repeat Step 2 until 'N' relations become one relation. If 'N' relations become single relation, then decomposition is called lossless, otherwise not.

Example : Consider a schema R(A,B,C,D,E,F,G,H,I,J) and functional dependencies.

F={AB→C, A→D, B→F, F→GH, D→IJ}. R is decomposed into

(i) R_i = (ABCDE)

R_j = (BFGH)

R_k = (DIJ)

Check whether the decomposition is lossless or not.

(ii) R_i = (ABCD)

R_j = (DE)

R_k = (BF)

R_l = (FGH)

R_m = (DIJ)

Check whether the decomposition is lossless or not.

Solution :

(i) Given

R_i = (ABCDE)

R_j = (BFGH)

R_k = (DIJ)

Step 1 :

$R_i \cup R_j \cup R_k$ = {(ABCDE) ∪ (BFGH) ∪ (DIJ)}

$= (ABCDEFGHIJ)$

$= R$

Step 1 satisfies the given condition, so it is true.

Step 2 :

For R_i and R_j:

$R_i \cap R_j = (ABCDE) \cap (BFGH)$

$= B$

Closure of B is as follows.

$B^+ = \{B, F, G, H\}$. Condition (ii) is satisfied, so R_i and R_j can be merged together.

Merging R_i and R_j we get,

$R_{ij} = (ABCDEFGH)$

Merge R_{ij} and R_k.

$R_{ij} \cap R_k = (ABCDEFGH) \cap (DIJ) = D$

Closure of D is as follows.

$D^+ = \{D, I, J\}$

Condition (ii) is satisfied, so R_{ij} and R_k can be merged together.

Merging R_{ij} and R_k we get,

$R_{ijk} = (ABCDEFGHIJ)$

$R_{ijk} = R$

Therefore, it is lossless decomposition.

(ii) Given

$R_i = (ABCD)$

$R_j = (DE)$

$R_k = (BF)$

$R_l = (FGH)$

$R_m = (DIJ)$

Step 1 :

$R_i \cup R_j \cup R_k \cup R_l \cup R_m = \{(ABCD) \cup (DE) \cup (BF) \cup (FGH) \cup (DIJ)\}$

$= (ABCDEFGHIJ)$

$= R$

Step 1 satisfies the given condition, so it is true.

Step 2 :

For R_i and R_j:

$R_i \cap R_j = (ABCD) \cap (DE) = D$

Closure of D is as follows.

$D^+ = \{D, I, J\}$

Condition (ii) is not satisfied, hence R_i and R_j cannot be merged together.

For R_i and R_k

$R_i \cap R_k = (ABCD) \cap (BF) = B$

Closure of B is as follows.

$B^+ = \{B, F, G, H\}$

Condition (ii) is satisfied, hence R_i and R_k can be merged together.

Merging R_i and R_k we get,

$R_{ik} = (ABCDF)$

For R_{ik} and R_l

$R_{ik} \cap R_l = (ABCDF) \cap (FGH) = F$

Closure of F is as follows.

$F^+ = \{F, G, H\}$

Condition (ii) is satisfied, hence R_{ik} and R_l can be merged together.

Merging R_{ik} and R_l we get,

$R_{ikl} = (ABCDFGH)$

For R_{ikl} and R_m

$R_{ikl} \cap R_m = (ABCDFGH) \cap (DIJ) = D$

Closure of D is as follows.

$D^+ = \{D, I, J\}$

Condition (ii) is satisfied, hence R_{ikl} and R_m can be merged together.

Merging R_{ikl} and R_m we get,

$R_{iklm} = (ABCDFGHIJ)$

For R_{iklm} and R_j

$R_{iklm} \cap R_j = (ABCDFGHIJ) \cap (DE) = D$

Closure of D is as follows.

$D^+ = \{D, I, J\}$

Condition (ii) of Step 2 does not satisfy, so R_{iklm} and R_j cannot be merged together. We are left with two decompositions which cannot be merged further into a single relation. So Step 3 condition does not satisfy. Therefore it is not lossless decomposition.

ii. Dependency preserving decomposition

Definition :

- Given a relation schema R (S, F) where F is a set of all FDs present in R on the attributes in S, R is decomposed into the relations schemas R_1, R_2 ..... R_n with FDs, F_1, F_2, ... F_n.
- Then the decomposition is dependency preserving if the closure of F' (where $F' = F_1 \cup F_2 \ldots\ldots \cup F_n$) is equal to F^+ i.e. $F'^+ = F^+$.

Example : Consider a schema R(A,B,C,D) and F = {A→BC}

R1 = (ABC) R2=(AD) Check whether the decomposition is dependency preserving or not.

Solution : The above decomposition is dependency preserving because Functional DependencyA→BC is a part of R1(ABC)

Example : Consider a schema R(A,B,C,D) and F = {A→B, C→D}

R1 = (AB) R2=(CD) Check whether the decomposition is lossless join and dependency preserving or not.

Solution : R1 ∩ R2 = ∅ which violates the condition of lossless join decomposition. Hence the decomposition is not lossless.

For dependency preserving, A→B can be ensured in R1(AB) and C→D can be ensured in R2(CD).

Hence, it is dependency preserving decomposition.

Example of Loss-less and Dependency Preserving :

- Given R (A, B, C, D) with the FDs F = {A → B, A → C, C→ D}, consider the decomposition R into R_1 = (A, B, C) with FDs, F_1 = {A →B, A→ C} and R_2 = {C, D} with FD's F_2 = {C → D}.
- In this decomposition all the original FDs can be logically derived from F_1 and F_2, i.e. F' = $F_1 \cup F_2$ = {A →B, A→C, C→D}. Hence, the decomposition is dependency preserving.
- Also, the common attribute 'C' forms a key of R_2. The decomposition of R into R_1 and R_2 is loss-less and dependency preserving.

Example of Lossy and Not Dependency Preserving:

- Given R = (A, B, C, D) with the FDs F = {A→B, A→C, A→D}, the decomposition of R into R_1 = (A, B, D) with the FDs, F_1 = {A→B, A→D} and R_2 = (B, C) with FDs F_2 = { } is lossy because the common attribute B is not a candidate key of either R_1 or R_2.
- In addition, the FD A→C is not implied by any FDs in R_1 and R_2. Thus, the decomposition is not dependency preserving and not loss-less.

6.7 CONCEPT OF NORMALIZATION (S-17), (W-17)

- Normalization is the process of distilling the structure of database. It removes repeated group of data into separate tables. Using normalization, we start with a collection of tables, apply sets of rules to eliminate anomalies, and produce a new collection of problem-free tables. The sets of rules are called normal forms.

6.7.1 First Normal Form (1NF)

- The 1NF states that every tuple must contain exactly one value for each attributes from the domain of that attribute. That is, multi-valued attributes, composite attributes and their combinations are not allowed in 1NF. A relation in First Normal Form is without internal repeating groups.
- **Definition :** A relation schema R is said to be in the 1NF if and only if the domains of all attributes of R contain atomic (or indivisible) values only.

6.7.2 Second Normal Form (2NF)

- The definition of 2NF is based on the concept of full functional dependency.
- **Definition :** A relational schema R is said to be in the 2NF, if every non-key attribute A in R is fully functionally dependent on the primary key. An attribute is non-key or non-prime attribute, if it is not a part of the candidate key of R.

6.7.3 Third Normal Form (3NF) (S-18)

- The normal form that is based on the concept of transitive dependency is the 3NF.
- A relation is in 3NF when it is in 2NF (i.e. fully functionally dependent on every Primary Key of relation) and no non-key attribute in relation is transitively dependent on the primary key.
- **Definition :** A relation schema R with a set F of functional dependencies is in the 3NF, if, for every FD $X \rightarrow Y$ in F, where X is any subset of attributes of R and Y is any single attribute of R, at least one of the following statements hold :
  1. $X \rightarrow Y$ is a trivial FD i.e., $Y \subseteq X$.
  2. X is a super key for R.
  3. Y is contained in a candidate key for R.

6.7.4 Boyce Codd Normal Form (BCNF)

- A relationship is said to be in BCNF if it is already in 3NF and the left hand side of every dependency is a candidate key.
- The BCNF was proposed to deal with the relations having two or more candidate keys that are composite and that overlap.
- **Definition :** A relation schema R is in a BCNF if for every Functional dependency $X \rightarrow A$ in F, where X is the subset of the attributes of R, and A is an attributes of R, one of the following statements holds :
  1. $X \rightarrow Y$ is a trivial FD, i.e., $Y \subseteq X$.
  2. X is a super key.

Example : Consider a relation Result (Seat_No, Email_id, Sub_Code, Marks).

In this relation, there are two candidate keys, namely 1. {Seat_No, Sub_Code} 2. {Email_id, Sub_Code}. These candidate keys are overlapped because they have Sub_Code as a common attribute. Therefore the above relation is in 3NF, but not in BCNF. To convert it into BCNF, we have to split the relation in two relations as follows:

1. Result (Seat_No, Sub_Code, Marks)
2. Stud_Info(Seat_No, Email_id)

Now the above relations are in BCNF.

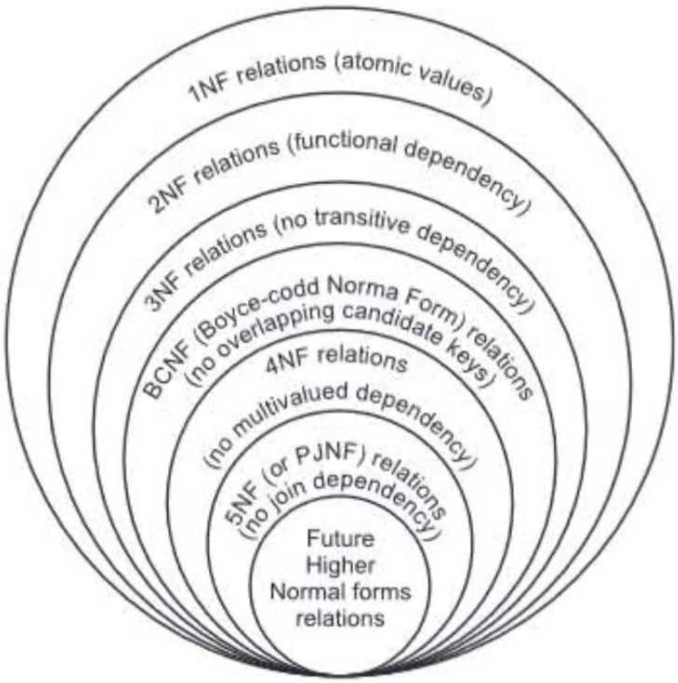


Fig. 6.7: Normal forms

6.8 EXAMPLES ON NORMALIZATION

Example 1 : Normalize the following table upto 3 NF. The database is related to order-report shown in the following table.

Order_no	Order_date	Invoice_no	Invoice_date	Product_no	Product_ description	Quantity_ ordered
25	17/3/19	1231	30/4/19	1	Computer	5
130	17/3/19	1232	30/4/19	2	Printer	5
520	28/3/19	1233	30/4/19	2	Printer	1
				4	A4 pages	10
				5	Main sheets	70
				6	Mouse	10

Solution:

The given date is in unnormalized form. First convert it into 1NF.

1NF :

No repeating groups. For this, partition each data structure containing repeating groups which accomplishes the same purpose. Fig. 6.7 is the result of 1NF.

Table 6.7 : Result of 1NF

Order_no	Order_date	Invoice_no	Invoice_date	Product_no	Product_ description	Quantity_ ordered
25	17/3/19	1231	30/4/19	1	Computer	5
130	17/3/19	1232	30/4/19	2	Printer	5
520	28/3/19	1233	30/4/19	2	Printer	1
520	28/3/19	1233	30/4/19	4	A4 pages	10
520	28/3/19	1233	30/4/19	5	Main sheets	70
520	28/3/19	1233	30/4/19	6	Mouse	10

- We remove the four repeating elements and group them into a new table called order_record, [See Table 6.8 (a)]. Rests of the elements are added into a new table called order-item. [See Table 6.8 (b)].

Table 6.8 (a) : Order_record

Order _ no	Order _date	Invoice _no	Invoice _date
25	17/3/19	1231	30/4/19
130	17/3/19	1232	30/4/19
520	28/3/19	1233	30/4/19

Table 6.8 (b): Order_item

Order _no	Product _no	Product _ description	Quantity_ ordered
25	1	Computer	5
130	2	Printer	5
520	2	Printer	1
520	4	A4 pages	10
520	5	Mainsheets	70
520	6	Mouse	10

- Both the relations Order_record and Order_item are in 1NF. But Order_item is not in its ideal form. Some of the attributes are not functionally dependent on primary key [Order_no, Product_no].
- Some non-key attributes such as product name, quantity are dependent only on Product_no and not on the key which we have considered.

2NF :

- All the non-key fields are dependent on the primary key (composite key). For this, verify that each non-key field is in first normal form and fully functionally dependent on primary key. Product_no → Product_description is the functional dependency.
- To convert Order_item into 2 NF, we separate the table (decompose) such that the above functional dependency satisfies correctly. [See Table 6.9 (a), 6.9 (b), 6.9 (c)]

Table 6.9 (a) : Order_record

Order_no	Order_date	Invoice_no	Invoice_date
25	13/3/19	1231	30/4/19
130	17/3/19	1232	30/4/19
520	28/3/19	1233	30/4/19

Table 6.9 (b) : Order_item

Order_no	Product_no	Quantity_ordered
25	1	5
130	2	5
520	2	1
520	4	10
520	5	70
520	6	10

Table 6.9 (c) : Product

Product_no	Product_ description
1	Computer
2	Printer
4	A4 pages
5	Mainsheets
6	Mouse

3NF :

When all non-key fields are independent of one another, then relation is in 3 NF. There is check for redundancy within the relation. Duplicate data elements which can be derived from other elements are removed at 3NF.

For example, in Order_record relation we have invoice date which is dependent on both i.e.Order_no and Invoice_no. To convert the Order_record into 3 NF we have to remove Invoice_date and group it with Invoice_no as the key in separate table.

So we have following 3NF relations : [See Table 6.10 (a), (b), (c), (d)]

Table 6.10(a) : Order_record

Order_no	Order_date	Invoice_no
25	17/3/19	1231
130	17/3/19	1232
520	28/3/19	1233

Table 6.10(b): Order_item

Order_no	Product_no	Quantity_ordered
25	1	5
130	2	5
520	2	1
520	4	10
520	5	70
520	6	10

Table 6.10(c) : Product

Table 6.10(d) : Invoice

Product_no	Product_description
1	Computer
2	Printer
4	A4 pages
5	Mainsheets
6	Mouse

Invoice_no	Invoice_date
1231	30/4/19
1232	30/4/19
1233	30/4/19

Therefore the relationship is transformed into following relations (3NF) is,

Table Name : Order_record

Order_no (primary key)
Order_date
Invoice_no (foreign key)

Table Name : Product

Product_no (primary key)
Product_description

Table Name : Order_Item

Order_no
Product_no (foreign key)
Quantity_ordered
Primary key : (order_no+product_no)

Table Name : Invoice

Invoice_no (primary key)
Invoice_date

Example 2 : Normalize the following table upto 3 NF. The database is related to cricket association.

The fields are –

Team_code skill_name club_city
player_code
team_name
player_name
player_age
player_skill_code
club_code
umpire_code
club_name
umpire_name
club_address
club_district

Following assumptions to be made –

1. A player can belong to one team and has only one skill.
2. An umpire can belong to one club only.
3. Club arranges matches between various teams and appoint umpires for the matches.

Solution: Unnormalized data:

club_code	team_code
club_name	team_name
club_address	player_code
club_district	player_name
club_city	player_age
umpire_code	player_skill_code
umpire_name	skill_name

1NF: Separate out the repeating group and find out key field.

Club table

club_code (primary key)
team_code (secondary key)
team_name
player_code
player_name
player_age
player_skill_code
skill_name

Umpire table

club_code (primary key)
club_name
club_address
club_district
club_city
umpire_code
umpire_name

2NF:

CT table

club_code
team_code
primary key : (club_code, team_code)

Team table

team_code (primary key)
team_name

Player table

team_code
player_code
player_name
player_age
player_skill_code
skill_name
primary key : (team_code, player_code)

Club table

club_code (primary key)
club_name
club_address
club_district
club_city

Umpire table

club_code
umpire_code (primary key)
umpire_name

3 NF: Find out the non-key elements depending upon some other non-key element.

Extract the elements which can be directly computed.

From the Player table player_age, player_skill_code, skill_name can be separated out as they are non key elements depend upon the non key element namely player_name. Similarly, club_name is a non key element.

player_info

player_name (primary key)
player_age
player_skill_code
skill_name

and

club_info

club_name (primary key)
club_address
club_district
club_city

The key elements from the new structures are found out. In the player_info table player_name is set out as a key field. In the club_info table, club_name is set out as a key field.

In the Player table team_code, player_code, player_name fields are present. In the Club table club_code, club_name fields are present.

Example 3 : Normalize the following table upto 3 NF.

ABC Traders			
cust_no : 001		order_ no : 1250	
cust_name : Shekhar		order_ date : 05/03/2019	
address : Camp		delivery_ date : 10/03/2019	
Item_no	**Description**	**Rate**	**Quantity**
001		Chair	65
014		Table	1200
015	Washing machine	14000	2
Remark : Delivery at London H.O.			

Solution:

Step 0: Firstly, we have to write down the data in the unnormalized form. For this, the data elements related to each other are to be listed out one after another.

Unnormalized data:

cust_no
cust_name
address
order_no
order_date
delivery_date
item no
desc
rate
quantity
remark

Step 1:

- To achieve 1NF, remove all the repeating groups.
- A repeating group is actually another relation. Hence, it is removed from the record and treated as an additional record structure or relation.
- In this example item_no, description, rate and quantity, etc. repeats with each customer. Hence, we repeat it.
- The main key from the original structure is to be brought into repeating group structure.
- The new structure is named with appropriate name.

Item table

cust_no (primary key)

item_no (partial key) (secondary)
desc
Rate
quantity

Customer table

cust_no (primary key)
cust_name
address
order_no
order_date
delivery_date
Remark

Step 2:

- As we know the above relation is in 1NF so we will go for 2NF we are looking for functional dependencies.
- For example, description of item is dependent on item_no, rate is dependent on item_no.
- Similarly, cust_name is dependent on cust_no, cust_address is dependent on cust_name and so on. We have to group such functionally dependent attributes and achieve 2NF.

So 2NF will have the following tables:

item_1 table

cust_no
order_no
item_no (primary key)
quantity

item_2 table

item_no (primary key)
desc
rate

cust_1 table

cust_no (primary key)
cust_name
address

cust_2 table

cust_no
order no (primary key)

cust_3 table

order_no (primary key)
order_date
delivery_date
remark

Step 3: In 3 NF we have to find out the non-key elements depending upon non-key elements. In the above structure address is a non-key element depends upon the cust_name which is also a non-key element. So, it can be defined as a new data structure and in which cust_name can be defined as a key field.

Example 4: Normalize the following data upto third normal form (3NF).

project_no
project_name
emp_no
emp_name
rate_category
hourly_rate

Solution: Above is the unnormalized data.

1 NF: Separate repeating groups

Employee_Project table

project_no
project_name
emp_no
emp_name
rate_category
hourly_rate

2 NF: Find the elements depending upon primary keys or partial keys.

Employee_Project Table

project_no (primary key)
emp_no (primary key)

Employee table

emp_no (primary key)
emp_name
rate_category
hourly_rate

Project table

project_no (primary key)
project_name

3 NF: Find non-key attributes. emp_name is not dependent on either rate_category or hourly_rate. Same is applied to rate_category. But, hourly_rate is dependent on rate_category.

Employe_Project table

project_no (primary key)

emp_no (primary key)

Employee table

emp_no (primary key)

emp_name

rate_category

Rate table

rate_category (primary key)

hourly_rate

Project table

project_no (primary key)

project_name

All above tables are in 3 NF.

Summary

- Normalization is a systematic process of organizing data efficiently in a database.
- A good database design ensures that the data that we want to store is consistent.
- Some pitfalls observed in relational database design are :
  - Redundancy.
  - Problem related with insertion of records.
  - Problem related with deletion of records.
  - Problem related with modification of records.
  - Inapplicable columns for most number of records.
- A functional dependency establishes a set of constraints among attributes.
- In a Functional Dependency $X \rightarrow Y$, X is known as determinant and Y is known as dependent.
- If the dependent is a subset of the determinant, trivial dependency exists.
- In a non-trivial dependency, the dependent (right hand side) part is not a subset of the determinant (left hand side) part.
- In transitive dependency a non prime attribute is determined by another non prime attribute.
- The six inference rules (Armstrong's Axioms) are: Reflexivity, Augmentation, Transitivity, Union, Decomposition, Pseudo transitivity.

- If closure of A (i.e. A^+) determines all the attributes of R, then A is called superkey.
- The difference between a superkey and a primary key is that a primary key is a minimal superkey.
- Goal of decomposition is to eliminate redundancy by decomposing a relation into multiple relations.
- The desirable properties of decomposition are : lossless join decomposition, dependency preserving decomposition.
- Normalization is the process of distilling the structure of database.
- Various normal forms : 1NF, 2NF, 3NF, BCNF, 4NF, 5NF
- 1NF : atomic values, 2NF: functional dependency, 3NF : no transitive dependency,
- BCNF : no overlapping of candidate keys, 4NF : no multivalued dependency, 5NF : no join dependency

Check Your Understanding

Q.1. Multiple Choice Questions:

1. Unnecessary repetition of data again and again in any relation is called as .....
   (a) dependency (b) redundancy
   (c) cardinality (d) none of these
2. A record is referred as .....
   (a) a criteria (b) a relation
   (c) a tuple (d) an attribute
3. Codd classified the normal form in the year .....
   (a) 1972 (b) 1973
   (c) 1960 (d) 1990
4. BCNF is more strict form than is .....
   (a) 1NF (b) 2NF
   (c) 3NF (d) 4NF
5. Which of the following normal form is related with the concept of atomic values?
   (a) 1NF (b) 2NF
   (c) 3NF (d) 4NF
6. Which of the following normal form is related with the concept of transitive dependency?
   (a) 1NF (b) 2NF
   (c) 3NF (d) 4NF
7. Which of the following normal form is related with the concept of overlapping of candidate keys?
   (a) 1NF (b) 2NF
   (c) 3NF (d) BCNF

8. In a Functional DependencyM→N, M is called as .....
   (a) determinant (b) dependent
   (c) attribute (d) function
9. Which of the following normal form is related with the concept of full functional dependency?
   (a) 1NF (b) 2NF
   (c) 3NF (d) BCNF
10. Multivalued dependencies helps in eliminating some forms of .....
   (a) redundancy (b) isolation
   (c) atomicity (d) inconsistency
11. A relational database consists of a collection of .....
   (a) tables (b) fields
   (c) records (d) keys
12. Database ..... which is the logical design of the database, and the database ..... is a snapshot of the data at a given time.
   (a) instance, schema (b) relation, schema
   (c) relation, domain (d) schema, instance
13. A domain of a column is atomic if the values in that column are .....
   (a) different (b) indivisible
   (c) constant (d) divisible
14. Which of the following attribute can be taken as a primary key?
   (a) emp_id (b) emp_name
   (c) emp_address (d) emp_city
15. The minimal set of superkey is called .....
   (a) schema key (b) candidate key
   (c) domain key (d) none of these
16. A relation is considered to be a .....
   (a) column (b) one dimensional table
   (c) two dimensional table (d) three dimensional table
17. Table is synonym for .....
   (a) field (b) record
   (c) relation (d) column
18. Key used to represent relationship between tables is called .....
   (a) primary key (b) secondary key
   (c) partial (d) foreign key

19. The relational model consists of .....
    (a) data in the form of tables (b) data redundancy
    (c) unorganized data (d) none of these
20. If no overlapping candidate keys exists in a relation, then it is in .....
    (a) 1NF (b) 2NF
    (c) 3NF (d) BCNF
21. For a regular entity .....
    (a) one relation is created. (b) two relations are created.
    (c) three relations are created. (d) none of these

Answers :

1 - (b)	2 - (c)	3 - (a)	4 - (c)	5 - (a)
6 - (c)	7 - (d)	8 - (a)	9 - (b)	10 - (a)
11 - (a)	12 - (d)	13 - (b)	14 - (a)	15 - (b)
16 - (c)	17 - (c)	18 - (d)	19 - (a)	20 - (d)
21 - (a)				

Q. 2. State True/False.

1. The goal of relational database design is to design tables that store information without unnecessary duplication of the data.
2. A good database design ensures that the data that we want to store is consistent.
3. Inserting many NULL values in the inapplicable columns does not result in wastage of storage space.
4. In M→N, M is called determinant and N is called dependent.
5. Transitive dependency is one type of functional dependency in which a non prime attribute is determined by another non prime attribute.
6. A relation in First Normal Form is without internal repeating groups.
7. A relational schema R is said to be in the 2NF, if every non-key attribute A in R is fully functionally dependent on the primary key.
8. A relation is in 3NF when it is in 2NF and no non-key attribute in relation is transitively dependent on the primary key.
9. A relation is in BCNF if there are no overlapping candidate keys in a relation.

Answers:

1 - True	2 - True	3 - False	4 - True
5 - True	6 - True	7 - True	8 - True
9 - True			

Practice Questions

Q.1. Answer the following questions in brief.

1. What is a functional dependency?
2. What is normalization?
3. What is decomposition in DBMS?
4. List the different pitfalls in relational database design.
5. List the desirable properties of decomposition.
6. What are the goals of decomposition?
7. What is full functional dependency?
8. In which normal form concept of transitive dependency is used?
9. In which normal form concept of overlapping candidate key is used?

Q.2. Answer the following questions.

1. Explain in detail about desirable properties of decomposition.
2. Explain in detail about the pitfalls in relational database design.
3. Consider the following unnormalized data
   Cust_no, cust_name, city, branch, br_no, loan_no, loan_amt, assets
   Normalize the above data to 3NF.
4. Consider the following unnormalized data:
   Roll_no, name, address, phone_no, class_code, class_name, sub_code, sub_name
   Normalize the above data to 3NF.
5. Describe the Armstrong's axioms.
6. Define BCNF. Explain how it is different from 3NF.
7. Describe about the difference between 1NF and 2NF relation.
8. Explain the difference between 2NF and 3NF relation.
9. How BCNF is more strict than 3NF? Comment.

Q. 3 Define the terms.

(a) 1NF
(b) 2NF
(c) 3NF
(d) BCNF
(e) Normalization
(f) functional dependency
(g) transitive dependency
(h) partial dependency
(i) super key
(j) primary key
(k) multivalued dependency

Previous Exams Questions

Summer 2017

1. Repetition of data in a table is known as ..... . [1 M]
   (i) dependency
   (ii) accuracy
   (iii) tendancy
   (iv) redundancy

Ans.: Refer to Section 6.2.

2. State the desirable properties of decomposition.

Ans.: Refer to Section 6.5

3. Define functional dependency.

Ans.: Refer to Section 6.3.

4. What is trivial functional dependency? Explain with example.

Ans.: Refer to Section 6.3.1.

5. What are the pitfalls in relational database design.

Ans.: Refer to Section 6.2.

6. Compute $(BCD)^+$ with the functional dependency:
   $A \rightarrow BC$, $CD \rightarrow E$, $E \rightarrow C$, $D \rightarrow AEH$,
   $ABH \rightarrow BD$, $DH \rightarrow BC$.

Ans.: Refer to Section 6.6

7. Explain the purpose of normalization and define the terms:
   (i) 1NF,
   (ii) 2NF,
   (iii) 3NF.

Ans.: Refer to Section 6.7.

Winter 2017

1. The meaning of the notation $X \rightarrow Y$ is ..... . [1 M]
   (i) X functionally determines Y
   (ii) Y functionally determines X
   (iii) Both (a) and (b)
   (iv) None of the above

Ans.: Refer to Section 6.3.

2. State the purpose of normalization.

Ans.: Refer to Section 6.7.

3. Let

$$R = \{A, B, C, D, E, F\}$$

and a set of FD's:

$$A \rightarrow BC, E \rightarrow CF, B \rightarrow E, CD \rightarrow EF, F \rightarrow D$$

Compute the closure of a set of attribute {A, B} under the given set of FDs.

Ans.: Refer to Section 6.3.2.

4. What are pitfalls in relational database design?

Ans.: Refer to Section 6.2.

5. What is normalization? Define the terms:

(i) 1 NF, (ii) 2NF, (ii) 3NF

Ans.: Refer to Section 6.7.

6. Write a short note on desirable properties of decomposition.

Ans.: Refer to Section 6.6.

Summer 2018

1. If $A \rightarrow C, CD \rightarrow E$ the $A \rightarrow E$ is , ... **[1 M]**

(i) transitivity rule (ii) pseudo transitivity rule

(iii) union rule (iv) none

Ans.: Refer to Section 6.6.

2. What do you mean by decomposition?

Ans.: Refer to Section 6.5.

3. Consider, R (A, B, C, D, G, H, I) and Set of functional dependency are

$F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$

Compute F^+.

Ans.: Refer to Section 6.3.

4. Consider the following:

Wholesaler (wno, wname, addr, city)

Product (Pno, Pname)

Wholesaler and product are related with m-m. Create a relational dB for above and convert it in 3NF and solve following queries.

(i) List all the wholesaler of product books.

(ii) Count the number at wholesaler in city Mumbai.

(iii) To print wholesalerwise product.

Ans.: Refer to Section 6.7.

5. Consider the following relation.
   Person (pnumber, pname, birthdate, in come)
   Area (aname, area-type)
   Area can have 1/more person living in it but person belongs to exactly one area.
   An attribute area type can have value urban/rural. Convert dB into 3NF and solve.
   (i) List names of all people in rural area.
   (ii) Give count of people whose income is below 30,000.
   (iii) List names of all people whose birthday is in January.

Ans.: Refer to Section 6.7.

6. Explain the 3NF algorithm.

Ans.: Refer to Section 6.7.3.

7. Consider, R = {A, B, C, D, E, F, G, H}
   F = {A → D, AB → DE, CE → G, E → H}
   Determine AB^+ and CF^+

Ans.: Refer to Section 6.3

8. Explain pitfalls in RDBMS.

Ans.: Refer to Section 6.2.

❖❖❖