DATABASE ENGINEERING

Unit 1

Question Bank — Introduction to DBMS and the ER Data Model

Section A — 37 short-answer questions (2 marks each)

Section B — 14 medium-answer questions (5 marks each)

Section C — 17 long-answer questions (7 marks each), with ER diagrams

Merges the Unit I syllabus topics, the model question bank and every Unit I question from PYQ R25, B26 and R24

Compiled from the official five-unit syllabus, the previous-year question papers (R25, B26, R24), the class slide decks, the model question banks and the handwritten notes. Every question is answered in full; differences are shown in two-column tables and ER diagrams are drawn wherever a question needs one.

Section A — 2-Mark Questions

Q1. Define data, database and DBMS.

Data is a collection of raw, unprocessed facts that has no meaning on its own — for example, the numbers 72 and 85. **Database** is an organised collection of related data stored so that it can be accessed, managed and updated efficiently. **DBMS** (Database Management System) is the software that creates, stores, retrieves and controls access to that database — for example MySQL, Oracle, SQL Server and PostgreSQL.

Q2. Differentiate between data and information.

Data	Information
Raw, unprocessed facts	Processed and organised data
No meaning by itself	Has meaning and is useful for decision making
Example: 72, 85, 90	Example: "Average marks = 82.3"
Input to the system	Output of the system

Q3. What is metadata? Give an example.

Metadata is data about data — it describes the structure, meaning and constraints of the actual data. It records the name of each table, its columns, data types, sizes, primary keys and access rights. Example: for a table STUDENT(SID, Name, Dept), the metadata records that SID is CHAR(5), Name is VARCHAR(30), SID is the primary key and Dept is a foreign key to DEPARTMENT. Metadata is stored in the **data dictionary**.

Q4. Define data dictionary and state its purpose.

PYQ R24 Q1(a)

A **data dictionary** (system catalog) is a centralised repository maintained by the DBMS that stores metadata about the database — relation names, attributes, data types, constraints, keys, indexes, users and access privileges. Its purpose is to let the DBMS and the DBA know the exact structure of every object without reading the actual data, to enforce constraints automatically, and to support query compilation, access control and change management.

Q5. What is a database system? How is it different from a DBMS?

A **database system** is the complete environment — the database itself, the DBMS software, the application programs, the users and the procedures. A **DBMS** is only the software component of that environment. In short, the DBMS is one part of the database system.

Q6. What is data abstraction?

Data abstraction is the process of hiding the low-level storage details from users and exposing only the parts of the database they need. It lets a user work with a logical view of data without knowing how the data is physically stored, indexed or organised on disk. It is implemented through the three levels of the three-schema architecture.

Q7. Define data independence and name its two types.

PYQ B26 Q1(c)

Data independence is the ability to change the schema at one level of the database system without changing the schema at the next higher level. It has two types: **logical data independence** (change the conceptual schema without changing external schemas or application

programs) and **physical data independence** (change the internal/physical schema without changing the conceptual schema).

Q8. Distinguish between physical and logical data independence.

Physical Data Independence	Logical Data Independence
Ability to change the internal (storage) level without changing the conceptual level	Ability to change the conceptual level without changing the external level
Easier to achieve	Harder to achieve
Example: changing a file organisation from sequential to hashed, adding an index, moving data to a new disk	Example: adding a new attribute or a new relation, splitting a relation, changing constraints
Sits between conceptual and internal levels	Sits between external and conceptual levels

Q9. What is a data model? Name four types.

PYQ model questions

A **data model** is a collection of concepts used to describe the structure of a database — the data types, the relationships among them, the constraints and the operations allowed. The four types are: **object-based logical models** (ER model, object-oriented model), **record-based logical models** (relational, network, hierarchical), **object-relational model** and **physical data models**.

Q10. Define entity and entity set with an example.

An **entity** is a real-world object or thing that has an independent existence — for example the student Asha, or the department CSE. An **entity set** is the collection of all entities of the same type, such as the set of all students, written STUDENT.

Q11. What is a weak entity set?

PYQ R25 Q1(b), B26 Q1(e)

A **weak entity set** is an entity set that does not have a key attribute of its own and therefore cannot be identified uniquely by its own attributes. It is identified through its owner entity set by means of an **identifying relationship** with total participation, and it uses a **partial key** together with the owner's primary key. Example: DEPENDENT of EMPLOYEE, or PAYMENT of LOAN.

Q12. What is a composite attribute? Give an example.

PYQ B26 Q1(d), R24 Q1(b)

A **composite attribute** is an attribute that can be divided into smaller sub-parts, each with its own independent meaning. Example: Address = (House_no, Street, City, PIN); Name = (First_name, Last_name). In the relational model only the simple components are stored.

Q13. Differentiate single-valued and multivalued, stored and derived attributes.

PYQ R25 Q1(f)

Type	Meaning	Example
Single-valued	Holds exactly one value for an entity	Date_of_birth, Roll_no
Multivalued	Holds a set of values for the same entity	Phone_no, Hobbies
Stored	Physically stored in the database	Date_of_birth, Salary
Derived	Computed from other stored attributes, not stored	Age (from DOB), Total (from Qty and Price)

Q14. What is a NULL value?

NULL represents the absence of a value — it means "unknown", "not applicable" or "not yet entered". It is not zero and not an empty string. Example: the Phone_no of a student who has not given a number. Primary key attributes cannot be NULL because that would violate entity integrity.

Q15. What is a key attribute?

A **key attribute** is an attribute (or set of attributes) whose values are distinct for every entity in the entity set, so it can identify each entity uniquely. It is shown underlined in an ER diagram. Example: Student_id in STUDENT, SSN in EMPLOYEE.

Q16. What is a relationship and a relationship set?

A **relationship** is an association among two or more entities — for example, the student Asha is enrolled in the course DBMS. A **relationship set** is the set of all such associations of the same type, such as ENROLLS between STUDENT and COURSE.

Q17. Define degree of a relationship.

PYQ R24 Q1(f)

The **degree** of a relationship is the number of entity sets that participate in it. A **unary** (degree 1) relationship relates an entity set to itself, a **binary** relationship (degree 2) relates two entity sets, and a **ternary** relationship (degree 3) relates three. Most practical relationships are binary.

Q18. List the different types of degrees of relationships.

PYQ B26 Q1(f)

Degree	Name	Example
1	Unary / recursive	An employee supervises another employee
2	Binary	A student enrolls in a course
3	Ternary	A supplier supplies a part to a project
n	n-ary	A general association among n entity sets

Q19. What is role in a relationship? What is a recursive relationship?

The **role** is the function that an entity plays in a relationship — for example in the relationship WORKS_FOR the employee plays the role of "worker" and the department plays the role of "employer". A **recursive relationship** is a relationship in which the same entity set participates more than once in different roles; role names are then essential to distinguish the two participations, as in EMPLOYEE supervises EMPLOYEE.

Q20. Define cardinality ratio and name its four types.

The **cardinality ratio** of a binary relationship specifies the maximum number of relationship instances in which an entity can participate. The four types are **1:1** (one to one), **1:N** (one to many), **N:1** (many to one) and **M:N** (many to many).

Q21. What is a participation constraint? Distinguish total from partial.

PYQ R25 model, B26 Q3(b)

The **participation constraint** specifies whether the existence of an entity depends on its participation in a relationship. **Total participation** means every entity of the set must participate (shown by a double line); **partial participation** means only some entities need participate (shown by a single line). Example: every LOAN must be related to a BRANCH (total), but not every EMPLOYEE need manage a department (partial).

Q22. Give an example of a one-to-many relationship.

PYQ R24 Q1(c)

DEPARTMENT — WORKS_FOR — EMPLOYEE. One department can have many employees, but each employee works for exactly one department. This is 1:N from DEPARTMENT to EMPLOYEE.

Q23. What is an identifying relationship? What is a partial key?

An **identifying relationship** is the relationship through which a weak entity set is identified by its owner entity set; it always has total participation on the weak side. A **partial key** (discriminator) is the attribute of the weak entity that distinguishes entities belonging to the same owner — for example Dependent_name for the dependents of one employee.

Q24. Define generalization and specialization.

Generalization is a bottom-up process that combines common attributes of several entity sets into a single higher-level superclass. **Specialization** is a top-down process that divides an entity set into subclasses on the basis of distinguishing attributes. Example: SECRETARY, ENGINEER and TECHNICIAN generalise into EMPLOYEE; EMPLOYEE specialises into those three.

Q25. What is aggregation in the ER model?

PYQ R24 Q1(d)

Aggregation is an abstraction in which a relationship set together with its participating entity sets is treated as a single higher-level entity set, so that it can participate in another relationship. Example: the relationship WORKS_ON between EMPLOYEE and PROJECT is aggregated so that a MANAGER can be associated with the whole works-on unit.

Q26. What is an EER model? Name its features.

PYQ R24 Q1(e)

The **Enhanced (Extended) ER model** adds to the basic ER model the concepts needed for more accurate schema design: **subclass and superclass**, **generalization and specialization**, **attribute inheritance**, **category or union type**, **aggregation**, and the **constraints** on specialization — disjoint or overlapping, and total or partial.

Q27. List the responsibilities of a Database Administrator.

PYQ R25 Q1(c), B26 Q1(b)

The DBA defines the conceptual, external and internal schemas; decides the storage structure and access method; grants and revokes user authorisation; monitors and tunes performance; plans and performs backup and recovery; ensures data integrity and security; and liaises with users to identify requirements and choose the DBMS.

Q28. Who are the users of a DBMS?

Naive end users (run pre-written forms and reports), **application programmers** (write programs that embed DML calls), **sophisticated users** (analysts who write their own queries), **the DBA** (controls the whole system) and **system analysts/designers** who define the requirements.

Q29. Name any two disadvantages of a file processing system.

PYQ R25 Q2(a), B26 Q2(b), R24 Q2(b)

Data redundancy and inconsistency — the same fact is stored in many files and repeated updates leave copies disagreeing; and **difficulty of concurrent access** — two users updating the same file at the same time can lose updates because there is no concurrency control. Other

disadvantages are program-data dependence, weak atomicity, poor security and limited data sharing.

Q30. Explain the term "exhaustive relationship" in an ER model.

PYQ R25 Q1(a) — wording in the paper is unclear; the two accepted readings are answered below

If the paper means an **exhaustiveness (completeness) constraint** on a specialization: it states that every entity of the superclass must belong to at least one subclass — a total specialization, as when every EMPLOYEE must be a secretary, engineer or technician. If the paper means an **exclusive (disjoint) relationship**: it states that an entity can belong to at most one subclass, so the subclasses do not overlap. Both are EER constraints and are usually written together as "disjoint, total" or "overlapping, partial". Note: some question banks use "exhaustive relationship" loosely for a **recursive relationship**, where an entity set is related to itself.

Q31. Write the fundamental differences between the network and hierarchical model.

Model question — Unit 1

Network Model	Hierarchical Model
Data organised as a graph	Data organised as a tree
A record may have several parents	Each child has exactly one parent
M:N relationships can be represented directly	M:N relationships cannot be represented directly
Owner and member records linked by sets	Parent and child records linked by pointers
Complex to design	Simple, rigid structure
Deleting an owner affects its members	Deleting a parent deletes all its children

Q32. Describe the differences between the ER model and the relational model.

Model question — Unit 1

ER Model	Relational Model
Conceptual design model	Implementation model
Entities, attributes, relationships, cardinality, participation	Tables, rows, columns, keys
Relationships drawn as diamonds	Relationships represented by foreign keys
No formal query language	Queried with relational algebra and SQL
Shows weak entities and total participation explicitly	Shows rules as key and referential constraints
Converted into tables by the mapping rules	Implemented directly in a DBMS

Q33. State the concept of data integrity.

Model question — Unit 1

Data integrity means the data in the database is accurate, consistent and valid at all times — it satisfies every rule the organisation has declared. It is maintained by domain constraints, key constraints, entity integrity and referential integrity, together with semantic rules enforced through assertions or triggers.

Q34. List any two steps for mapping an ER model into a relational model.

Model question — Unit 1

1. Each strong entity set becomes a table, with its key attribute as the primary key. 2. Each M:N relationship becomes a separate table whose primary key is the combination of the participating entity keys. Example: STUDENT and COURSE become two tables, and the M:N relationship ENROLLS becomes a third table ENROLLS(Student_id, Course_no, Grade).

Q35. Explain the significance of descriptive attributes and NULL attributes.

Model question — Unit 1

A **descriptive attribute** belongs to a relationship rather than to any single entity — Grade in ENROLLS, Hours in WORKS_ON. It cannot be moved into either entity table without losing information, and it is the reason an M:N relationship must become its own table. A **NULL** marks a value that is unknown, not applicable or not yet available; it allows optional facts to be stored without inventing dummy values, but it must be handled carefully because NULL never equals anything, aggregates ignore it, and joins drop it.

Q36. Compare composite and multivalued attributes.

Model question — Unit 1

Composite attribute	Multivalued attribute
Can be divided into sub-parts with independent meaning	Holds a set of values for the same entity
Drawn as one ellipse with the components attached	Drawn as a double ellipse
Maps to a single table, storing only the simple components	Maps to a separate table holding the key and the value
Example: Address (House_no, Street, City)	Example: Phone_no, Hobbies

Q37. List the features of the extended ER (EER) model.

Model question — Unit 1

Subclass and superclass; attribute inheritance; generalization and specialization; category or union type; aggregation; and the constraints on specialization — disjoint or overlapping, total or partial, and attribute-defined or user-defined.

Section B — 5-Mark Questions

Q1. Explain the disadvantages of a traditional file processing system.

5 marks

In a file processing system each application keeps its own private files and its own programs, so the same fact is stored in several places.

Disadvantage	What goes wrong
Data redundancy	The same fact is stored in many files, wasting storage
Data inconsistency	One copy is updated and another is not, so the files disagree
Program–data dependence	The file layout is written into every program, so any change forces rewrites
Difficulty of access	Only pre-written reports exist; a new question needs a new program
No concurrency control	Simultaneous updates can overwrite one another
Weak atomicity	A half-completed update leaves the data inconsistent
Poor security	Access control works per file, not per record or field
Limited sharing	Data stays isolated inside departments

Each of these problems is a direct consequence of having no central control over the data and no shared description of what the data means.

Q2. Explain the need for a DBMS.

5 marks

A DBMS is needed because it removes every limitation of the file system in a specific way.

- **One integrated database** — each fact is stored once, so redundancy and inconsistency disappear.
- **Schema in the data dictionary** — the description of the data is stored separately from the data, which gives data independence.
- **High-level query language** — SQL allows an unanticipated question to be answered without writing a new program.
- **Central constraint enforcement** — domain, key, entity and referential rules are declared once and checked by the DBMS.
- **Concurrency control** — transactions are isolated, so lost updates and dirty reads do not occur.
- **Recovery** — logs and checkpoints restore a consistent state after failure.
- **Security** — views and privileges restrict what each user may see and do.

In one sentence: a DBMS stores the data once, describes it in one place, controls who may do what, and guarantees that a transaction either completes fully or not at all.

Q3. Explain the functions of a DBMS.

5 marks

Function	What it does
Data definition	Creates and alters the schema using DDL
Data manipulation	Inserts, updates, deletes and retrieves data using DML and DQL
Storage management	Manages files, pages and indexes on disk
Security	Authenticates users and enforces privileges
Integrity enforcement	Checks domain, key, entity and referential constraints
Concurrency control	Keeps interleaved transactions correct
Recovery	Restores a consistent state after failure
Data dictionary management	Maintains all metadata
Utilities	Backup, restore, import, export, statistics, reorganisation

Q4. Explain the data dictionary and its contents.

5 marks

A data dictionary, also called the **system catalog**, is the centralised repository in which the DBMS stores metadata about the database.

What it contains

Relation names and their attributes; data types and sizes; primary, candidate and foreign keys; integrity constraints; indexes and access paths; views; users, privileges and ownership; storage details and statistics.

Why it is needed

The DBMS reads it to parse and validate every query, to enforce constraints without reading the data, and to check access rights. It also gives the DBA a single place to see and document the whole schema.

Definition for the exam

The data dictionary is the DBMS's own database of definitions — it stores descriptions of the data, never the data itself.

Q5. Differentiate logical and physical data independence with examples.

5 marks

Logical data independence	Physical data independence
Ability to change the conceptual schema without changing external schemas or programs	Ability to change the internal schema without changing the conceptual schema
Harder to achieve	Easier to achieve
Example: adding a new attribute or a new relation, splitting a relation, changing a constraint	Example: adding an index, changing the file organisation, moving data to a new disk
Sits between the external and conceptual levels	Sits between the conceptual and internal levels

Logical independence is harder because a change to the conceptual schema can remove something a view depends on. Physical changes can almost always be absorbed by the conceptual/internal mapping, which is why a program survives a change of storage device.

Q6. Explain the users of a DBMS.

5 marks

User	What they do
Naive end users	Invoke pre-written application programs through forms and reports; unaware of the database
Application programmers	Write programs in a host language with embedded DML calls
Sophisticated users	Analysts and scientists who write their own queries directly
DBA	Controls the schema, storage, authorisation, performance and recovery
Database designers	Identify the data to be stored and choose the structures and views

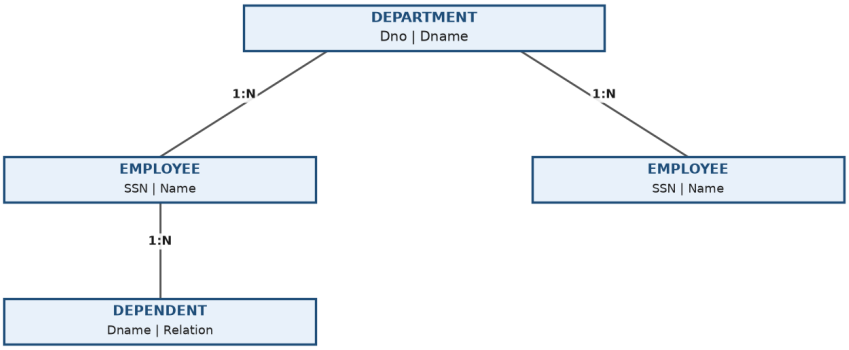
Each class needs a different level of the three-schema architecture. A naive user needs only an external view, a programmer needs the conceptual schema, and only the DBA works with the internal schema.

Q7. Explain the hierarchical model with its advantages and disadvantages.

5 marks

Data is organised as a **tree** of records: each parent may have many children, but each child has exactly one parent. Example: DEPARTMENT at the root, EMPLOYEE as children, DEPENDENT as grandchildren.

Hierarchical Model — a tree of records



Each child has exactly ONE parent — many-to-many is impossible

Advantages

Simple and fast for one-to-many navigation along the tree; storage is efficient because the structure is fixed; access along a known path is quick.

Disadvantages

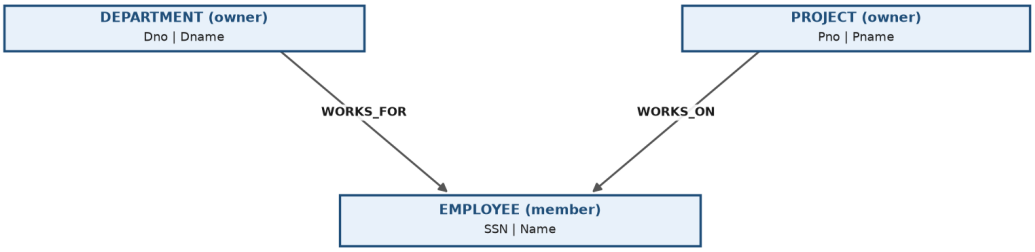
Problem	Consequence
Insertion anomaly	A child cannot be inserted without its parent
Deletion anomaly	Deleting a parent deletes all its children
Update anomaly	A repeated fact must be updated at every occurrence
Structural rigidity	M:N relationships cannot be represented directly
Navigational access	The programmer must know the tree structure

Q8. Explain the network model with its advantages and disadvantages.

5 marks

Data is organised as a **graph** of records connected by sets, so a record may have several parents. Each set has one **owner record** and one or more **member records**, linked by pointers.

Network Model — a graph of records linked by sets



A member record may have SEVERAL owners — many-to-many is possible

Advantages

Represents M:N relationships directly; less redundancy than the hierarchical model; efficient navigation once the pointers are known.

Disadvantages

The programmer must understand the physical pointer structure; any change to the set structure forces programs to be rewritten; the design is complex; deleting an owner affects its members.

Q9. Explain the characteristics of the relational model.

5 marks

Data is represented as **tables** of rows and columns, and relationships are represented by **common values** rather than pointers.

- Data is visible as relations, which is easy to understand.
- No pointers exist, so nothing needs maintaining when data moves.
- Any M:N relationship is expressed by an extra table.
- It has a sound theoretical base in set theory and functional dependencies.
- It is queried declaratively with SQL.

Relational Model — data as a table of rows and columns

degree = 4 (columns)

cardinality = 3 rows

SID	Name	Dept	CGPA
S1	Asha	CSE	8.7
S2	Bikash	AIDS	8.1
S3	Chitra	CSE	7.5

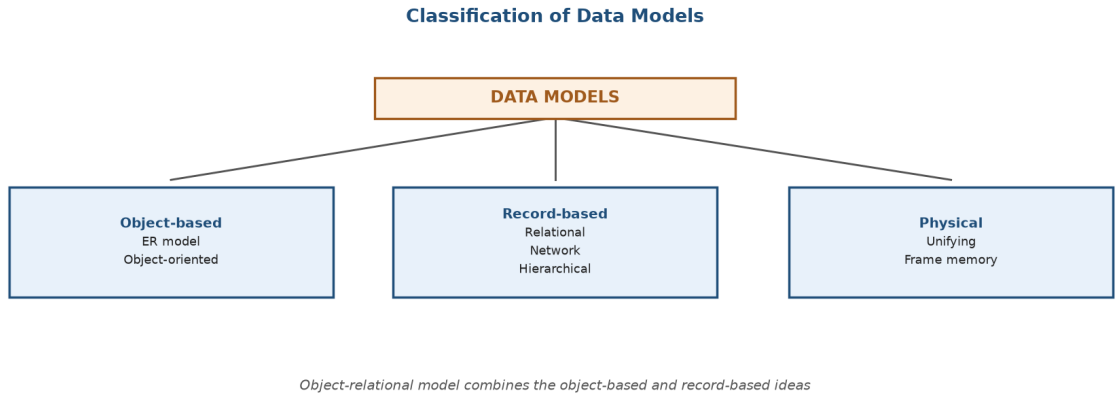
A relation is a set of tuples; each tuple is one entity

Because relationships are values and not addresses, the logical view is independent of storage, and normalisation theory then guarantees a design free of the anomalies that plagued the hierarchical and network models.

Q10. Explain the classification of data models.

5 marks

A data model is a collection of concepts used to describe the structure of a database — the data types, the relationships among them, the constraints, and the operations allowed.

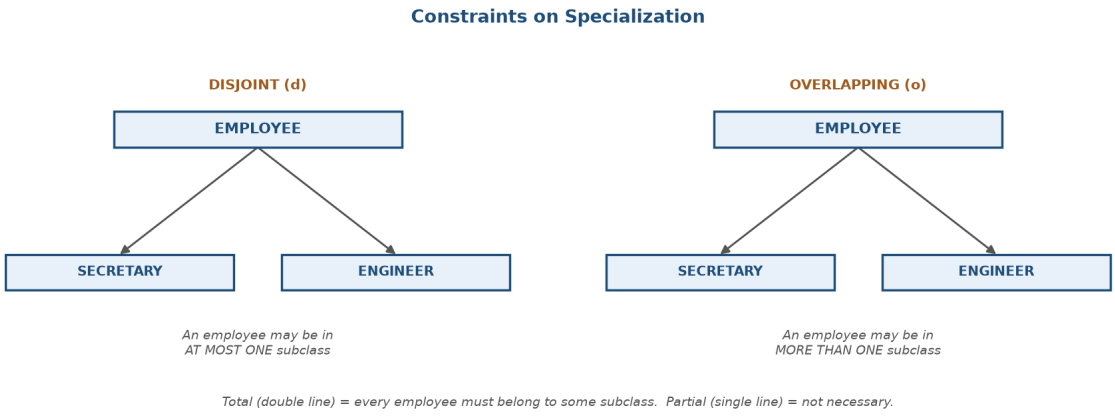


Object-relational model combines the object-based and record-based ideas

Category	Models	Idea
Object-based logical	ER model, object-oriented model	Describe data as entities and objects
Record-based logical	Relational, network, hierarchical	Describe data as fixed-format records
Object-relational	Extended relational	Add objects and inheritance to tables
Physical	Unifying, frame-memory	Describe how data is stored at the lowest level

Q11. Explain the constraints on generalization and specialization.

5 marks



Total (double line) = every employee must belong to some subclass. Partial (single line) = not necessary.

Constraint	Meaning
Disjoint	An entity belongs to at most one subclass (exclusive)
Overlapping	An entity may belong to more than one subclass
Total	Every superclass entity must belong to some subclass (exhaustive)
Partial	An entity need not belong to any subclass
Attribute-defined	Membership is decided by a specific attribute value
User-defined	Membership is decided by the user at insertion time

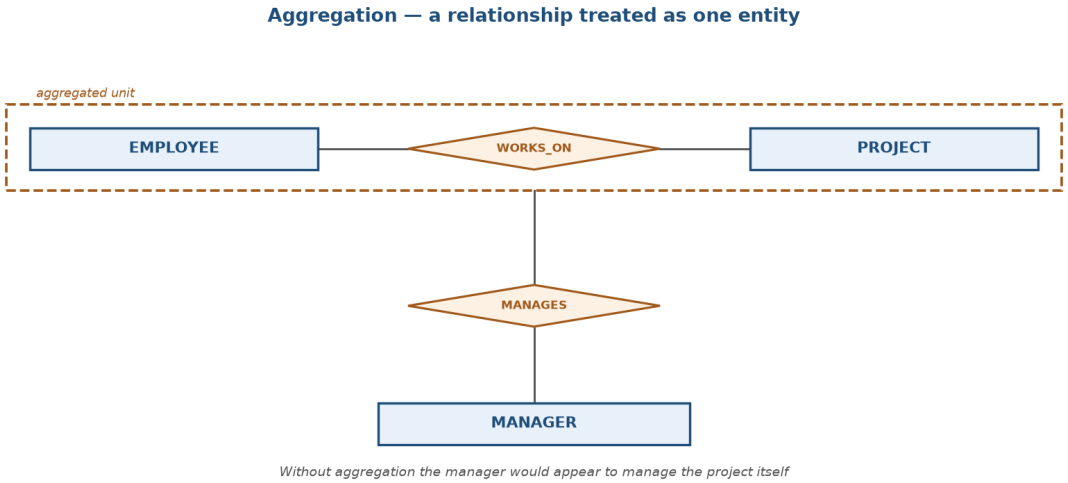
The constraint decides which mapping strategy is valid later. A total and disjoint specialization can be merged into a single table with a type attribute, whereas an overlapping one cannot be enforced that way without extra checks.

Q12. Explain aggregation and category (union type) in the EER model.

5 marks

Aggregation

Aggregation treats a **relationship set together with its participating entity sets** as a single higher-level entity, so that this unit can itself take part in another relationship.



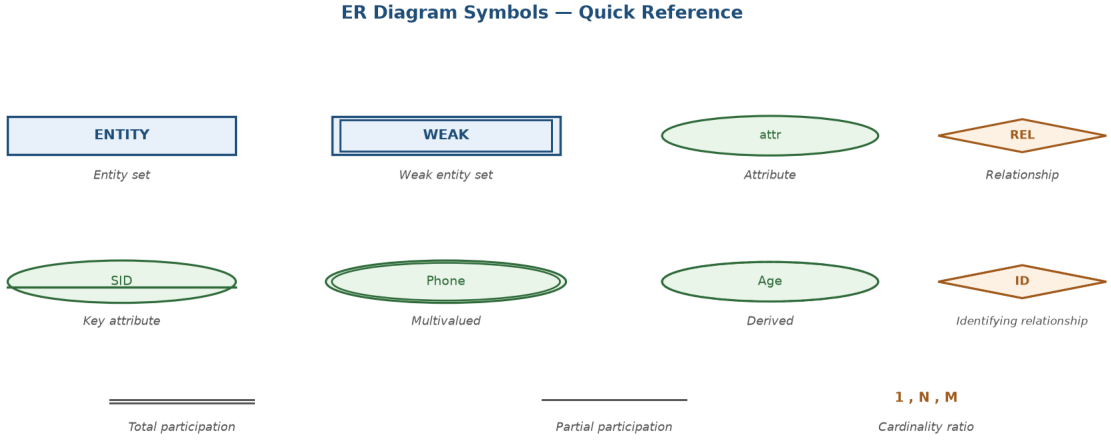
It is needed when a relationship must be related to another entity — for example an employee managing another employee's work on a project. Without aggregation the diagram is ambiguous, because the manager would appear to manage the project or the employee rather than the participation itself.

Category (union type)

A category is a subclass that is a subset of the **union** of several superclasses. Example: OWNER is a category whose members may be either a PERSON or a COMPANY, but not both.

Q13. Explain the ER diagram symbols.

5 marks



Symbol	Meaning
Rectangle	Entity set
Double rectangle	Weak entity set
Ellipse	Attribute

Symbol	Meaning
Underlined attribute	Key attribute
Double ellipse	Multivalued attribute
Dashed ellipse	Derived attribute
Diamond	Relationship set
Double diamond	Identifying relationship
Double line	Total participation
Single line	Partial participation
Label 1, N, M on the line	Cardinality ratio

Q14. Explain metadata and its role in a database system.

5 marks

Metadata is data about data — it describes the structure, meaning and constraints of the stored data without being the data itself.

What metadata records

The name of every table and its columns, the data type and size of each column, which attribute is the primary key, which are foreign keys, the constraints, the indexes, the views, and the users with their privileges.

Where it is kept

In the **data dictionary**, or system catalog, which the DBMS reads constantly.

Its role

- It lets the DBMS parse and validate a query without reading the data.
- It is how constraints and access rights are checked.
- It gives the DBA one place to see and document the schema.
- It is what makes data independence possible, because programs refer to the metadata rather than to the file layout.

Example: for STUDENT(SID, Name, Dept), the metadata records that SID is CHAR(5) and is the primary key, and that Dept is a foreign key to DEPARTMENT.

Section C — 7-Mark Questions

Q1. Explain the disadvantages of a traditional file processing system and the need for a DBMS.

PYQ R25 Q2(a), B26 Q2(b), R24 Q2(b)

In a file processing system each application keeps its own private files and its own programs to manipulate them. This design fails in the following ways.

1. Data redundancy and inconsistency

The same fact — say a student's address — is stored in the admission file, the examination file and the fee file. Storage is wasted, and when the address changes one copy may be updated while another is not, so the database becomes inconsistent.

2. Program-data dependence

The physical layout of the file is written into every application program. Adding a field, changing a record length or moving to a new device forces every program that touches that file to be rewritten. Data definition is buried in the program, not stored separately.

3. Difficulty of accessing data

A file system gives only fixed, pre-written reports. If a new question is asked — "list students of CSE with CGPA above 8" — a new program has to be written and run, because there is no general query facility.

4. No concurrent access control

When several users update the same file at the same time, one update can overwrite another (the lost-update problem) or a user may read a value that is later rolled back (dirty read). File systems have no transaction concept.

5. Weak atomicity and recovery

A transfer of money that debits one file and credits another can fail half-way, leaving the data inconsistent. There is no automatic rollback and no log-based recovery.

6. Poor security and limited sharing

Access control is at the level of whole files, not individual records or fields, so fine-grained authorisation is impossible. Data is also scattered across departments and cannot be shared easily.

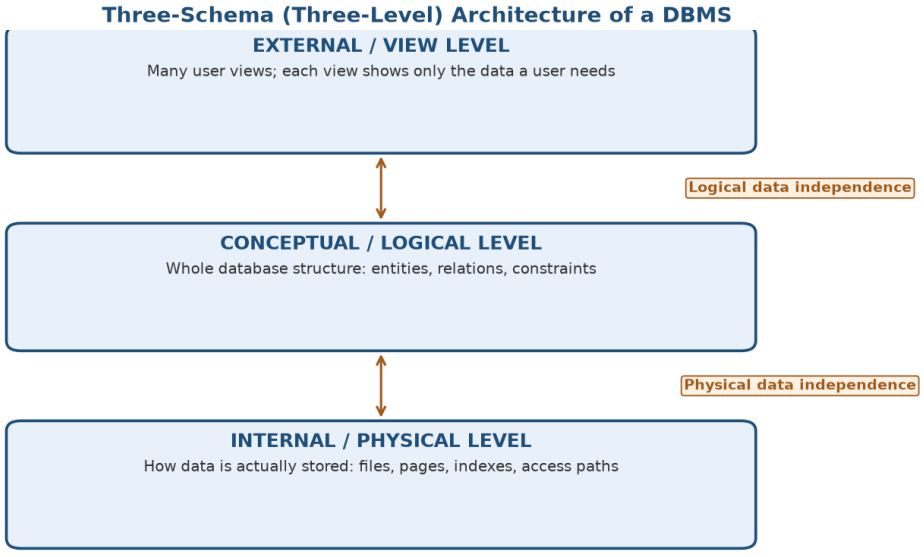
How a DBMS solves these

A DBMS stores a single integrated database described by a schema held in the data dictionary, so each fact is stored once. It separates programs from data through the three-schema architecture and data independence, provides a high-level query language (SQL), enforces integrity constraints centrally, provides concurrency control and recovery through transactions with ACID properties, and supports granular authorisation through views and privileges.

Q2. Illustrate the three-schema levels of database architecture and explain how data abstraction helps.

PYQ R25 Q2(c), R24 Q2(a)

The three-schema architecture, also called the three-level architecture, separates the database description into three levels so that each kind of user sees only what concerns them.



External/Conceptual mapping and Conceptual/Internal mapping isolate each level from changes in the level below.

External (view) level

The highest level, closest to the users. Each user or application sees only its own **external schema** — a tailored view of the part of the database it needs, with irrelevant attributes hidden and some

values possibly derived. A database may have many external schemas.

Conceptual (logical) level

The middle level, describing **what** data is stored and how it is related: all entity sets, their attributes, the relationships, the constraints and the security rules. It is a complete but implementation-independent description, usually prepared in the ER or relational model.

Internal (physical) level

The lowest level, describing **how** the data is actually stored: file organisations, record layouts, page sizes, indexes, access paths and storage allocation.

Mappings and data independence

The **external/conceptual mapping** defines how each view is derived from the conceptual schema, and the **conceptual/internal mapping** defines how the conceptual schema is realised on disk. Because of these mappings, the conceptual schema can change without touching the views (logical data independence) and the internal schema can change without touching the conceptual schema (physical data independence).

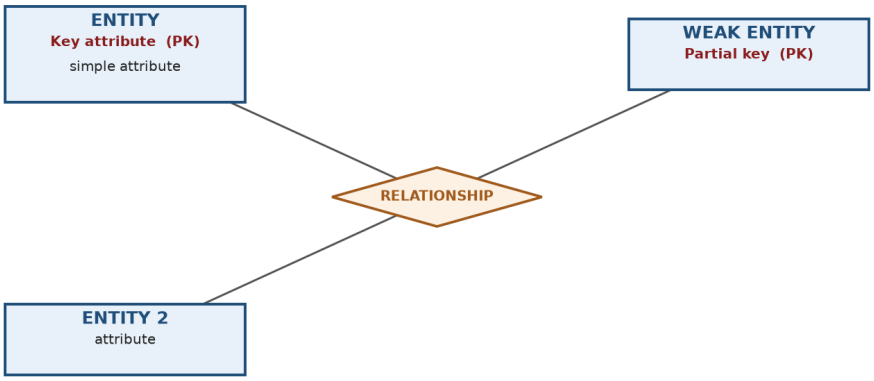
How data abstraction helps

Data abstraction hides the complexity of the lower levels from the higher ones. A naive user sees only a form, an application programmer sees the conceptual schema, and only the DBA sees the internal schema. This makes the system easier to understand and to use, protects programs from storage changes, allows the same data to be presented differently to different users, and improves security because a user can be restricted to a view.

Q3. Explain the major components of the ER model: entities, attributes, relationships and keys, with relationship types.

PYQ R25 Q2(b)

ER Model — Basic Notation Used in These Notes



Bold attribute with (PK) = key attribute. Weak entity is drawn in a double-bordered box in a full ER diagram.

Entity and entity set

An **entity** is a distinguishable real-world object; an **entity set** is a collection of entities of the same type. Entity sets are drawn as rectangles.

Attributes

Attributes describe the properties of an entity. They may be **simple** (atomic, e.g. Age) or **composite** (divisible, e.g. Address), **single-valued** (Date_of_birth) or **multivalued** (Phone_no), **stored** (Date_of_birth) or **derived** (Age). A **key attribute** uniquely identifies an entity and is underlined.

Relationships and relationship sets

A **relationship** associates entities; a **relationship set** is a set of relationships of the same type, drawn as a diamond. Relationships are labelled with the roles the entities play when the same entity set participates twice.

Types of relationships

Basis	Types	Meaning
Degree	Unary, binary, ternary, n-ary	Number of participating entity sets
Cardinality ratio	1:1, 1:N, N:1, M:N	Maximum instances per entity
Participation	Total, partial	Whether participation is compulsory
Direction	Recursive, non-recursive	Same entity set participating twice or not

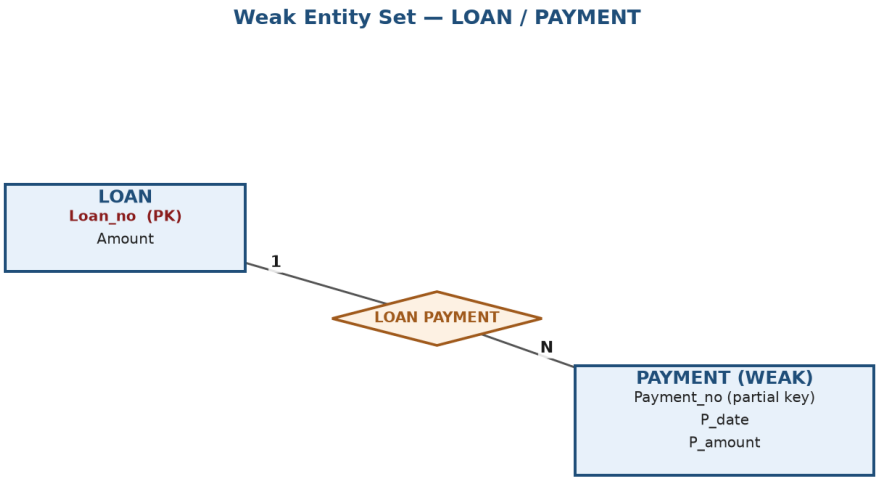
Keys

The **primary key** identifies an entity uniquely. A **partial key** (discriminator) is used in a weak entity set together with the owner's key. Keys are what allow an ER diagram to be converted into relational tables later.

Q4. Compare strong and weak entity sets, with an example. Explain identifying relationship and partial key.

PYQ R25 Q1(b), B26 Q1(e), R24 Q2(c)

Strong Entity Set	Weak Entity Set
Has its own key attribute	Has no key of its own; only a partial key
Existence is independent of other entity sets	Existence depends on an owner entity set
Drawn with a single rectangle	Drawn with a double rectangle
Participation in relationships may be partial	Total participation in the identifying relationship
Mapped to a table with its own primary key	Mapped to a table whose primary key is owner key + partial key
Example: EMPLOYEE, STUDENT, LOAN	Example: DEPENDENT of EMPLOYEE, PAYMENT of LOAN, SECTION of COURSE



PAYMENT has no key of its own; it is identified by Loan_no + Payment_no and exists only with total participation in LOAN_PAYMENT.

Identifying relationship

The relationship that links a weak entity set to its owner is called the **identifying relationship** and is drawn with a double diamond. It always has total participation on the weak side, because a weak entity cannot exist without its owner.

Partial key

The **partial key** (also called the discriminator) is the attribute of the weak entity set that distinguishes entities belonging to the same owner. In LOAN–PAYMENT, Payment_no alone repeats across loans, but the combination (Loan_no, Payment_no) is unique, so Payment_no is the partial key and the full key is formed with the owner's key.

Student–Dependent example

DEPENDENT has attributes Dependent_name, Sex, Birthdate and Relationship. Dependent_name is only a partial key: two employees may both have a son named "Rahul". The weak entity is therefore identified by (SSN of the employee, Dependent_name).

Q5. Explain generalization, specialization and aggregation with suitable diagrams.

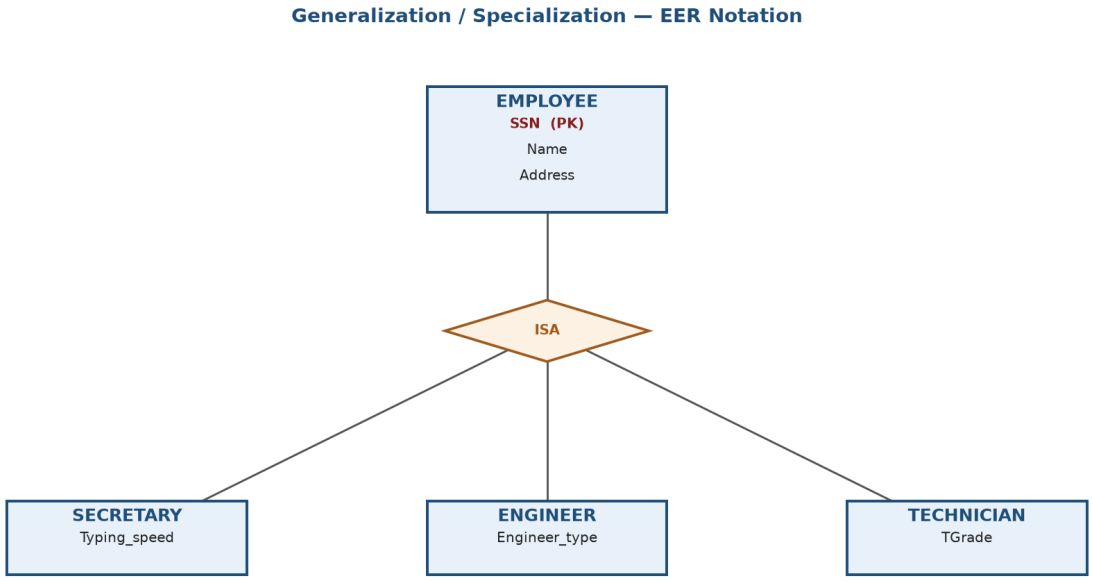
PYQ R25 Q3(a), R24 Q1(d)

Generalization

Generalization is a **bottom-up** abstraction: several entity sets that share common attributes are combined into a single superclass, and the attributes common to all of them are attached to the superclass. Example: CAR and TRUCK generalise into VEHICLE; SECRETARY, ENGINEER and TECHNICIAN generalise into EMPLOYEE.

Specialization

Specialization is a **top-down** abstraction: an entity set is divided into subclasses on the basis of some distinguishing characteristic, and attributes specific to a subclass are attached only to that subclass. Example: EMPLOYEE specialises into SECRETARY (Typing_speed), ENGINEER (Engineer_type) and TECHNICIAN (TGrade).



ISA is the specialization/generalization relationship. Attributes are inherited downward; the constraint may be disjoint or overlapping, total or partial.

Comparison

Generalization	Specialization
Bottom-up process	Top-down process
Combines entity sets into a superclass	Divides a superclass into subclasses
Removes duplication of common attributes	Adds attributes specific to each subclass
Starts from specific entity sets	Starts from one general entity set

Attribute inheritance and constraints

A subclass inherits all attributes of its superclass and adds its own, and also inherits participation in the superclass's relationships. The specialization may be **disjoint** (an entity belongs to at most one subclass) or **overlapping**, and **total** (every superclass entity must belong to a subclass) or **partial**.

Aggregation

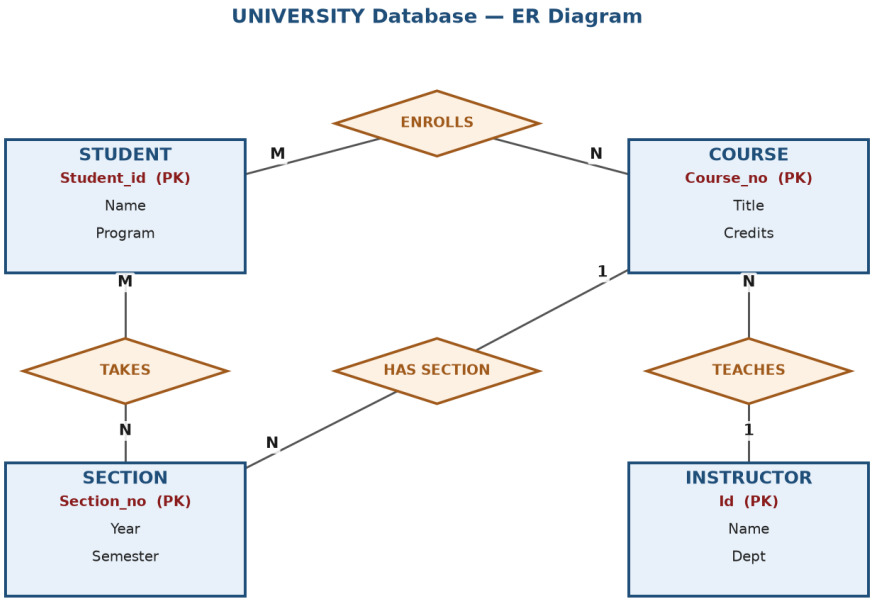
Aggregation treats a relationship set together with its participating entity sets as a single higher-level entity, so that this unit can take part in another relationship. It is needed when a relationship must be related to another entity — for example, an employee manages the works-on of another employee on a project. Without aggregation the ER diagram would be ambiguous, because the manager would appear to manage the project or the employee rather than the participation itself.

Q6. Design an ER diagram for a university database and map it to relational tables.

PYQ R25 Q3(b)

Problem

A university has students, courses and instructors. Each student can enroll in multiple courses and each course can be taken by many students. Each course is taught by one instructor. Attributes, relationships and key constraints must be shown, and the diagram must then be mapped to tables.



SECTION is a weak entity of COURSE; its partial key is Section_no. Grade is a descriptive attribute of TAKES.

Step 1 — Entity sets and keys

STUDENT with key Student_id; COURSE with key Course_no; INSTRUCTOR with key Id; SECTION, a weak entity of COURSE, with partial key Section_no.

Step 2 — Relationships and cardinalities

ENROLLS is M:N between STUDENT and COURSE. TEACHES is N:1 from COURSE to INSTRUCTOR, because one instructor teaches many courses but a course has one instructor. HAS_SECTION is 1:N from COURSE to SECTION with total participation on the SECTION side. TAKES is M:N between STUDENT and SECTION, carrying the descriptive attribute Grade.

Step 3 — Mapping to relational tables

Table	Attributes	Key and constraints
STUDENT	Student_id, Name, Program	PK: Student_id
COURSE	Course_no, Title, Credits, Id	PK: Course_no; FK: Id references INSTRUCTOR
INSTRUCTOR	Id, Name, Dept	PK: Id
SECTION	Course_no, Section_no, Year, Semester	PK: (Course_no, Section_no); FK: Course_no references COURSE
ENROLLS	Student_id, Course_no	PK: (Student_id, Course_no); both are FKs
TAKES	Student_id, Course_no, Section_no, Grade	PK: (Student_id, Course_no, Section_no); FKs to STUDENT and SECTION

Rules used

The M:N relationships ENROLLS and TAKES each become their own table with a composite primary key. The 1:N relationship TEACHES is merged into the many side, so the instructor's key becomes a foreign key inside COURSE. The weak entity SECTION receives the owner's key as part of its primary key.

Q7. Explain the different record-based data models with examples and state the advantages of the relational model.

PYQ model questions, B26 Q2(a)

Hierarchical model

Data is organised as a tree of records, each parent having many children but each child having exactly one parent. Example: DEPARTMENT at the root, EMPLOYEE as children, DEPENDENT as grandchildren. It is fast for one-to-many navigation along the tree, but it cannot represent many-to-many relationships directly, and it suffers from insertion, deletion and update anomalies because the tree structure is rigid.

Network model

Data is organised as a graph of records connected by sets, where a record can have several parents. The owner record and member records of a set are linked by pointers. It represents many-to-many relationships, but the programmer must know the physical pointer structure, so it is complex to design and to change.

Relational model

Data is represented as tables (relations) of rows and columns. There are no pointers; relationships are represented by common attribute values, that is by foreign keys. It is based on set theory and is queried with relational algebra, relational calculus and SQL.

Advantages of the relational model

Point	Reason
Simplicity	Data is visible as tables, easy to understand and to explain
Physical independence	No pointers to maintain; the logical view is independent of storage
Flexible relationships	Any M:N relationship is expressed by an extra table
Strong theoretical base	Normalisation, FD theory and closure give provable designs
Ad hoc querying	SQL allows unanticipated queries without new programs
Data integrity	Keys and constraints are declared once and enforced centrally

Q8. Explain the various types of attributes in the ER model with suitable examples.

PYQ R25 Q1(f), model questions

Type	Definition	Example
Simple	Atomic, cannot be divided	Age, Salary, Roll_no
Composite	Can be divided into sub-parts	Address (House_no, Street, City, PIN)
Single-valued	One value per entity	Date_of_birth, Gender
Multivalued	A set of values per entity	Phone_no, Hobbies, Qualifications
Stored	Physically stored	Date_of_birth, Basic_pay
Derived	Computed from other attributes	Age from DOB, Total from Qty and Rate
Key	Uniquely identifies the entity	Student_id, SSN
Partial key	Distinguishes entities of a weak set within one owner	Dependent_name
Descriptive	Belongs to a relationship, not an entity	Grade in ENROLLS, Hours in WORKS_ON
NULL	Value unknown or not applicable	Middle_name not given

Why descriptive attributes matter

A **descriptive attribute** describes the relationship instance rather than any one entity. Grade does not belong to the student alone or to the course alone; it belongs to the pair. When the relationship is mapped to a table, the descriptive attributes are stored in that table, and they are the reason an

M:N relationship cannot be merged into either entity table.

Q9. Explain cardinality ratios and participation constraints with an ER diagram.

PYQ B26 Q3(b)

Cardinality ratio

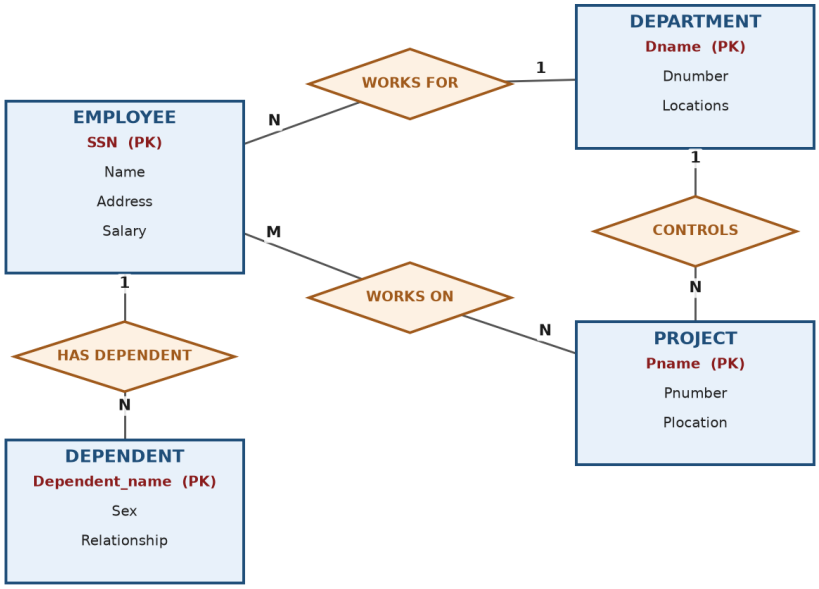
The cardinality ratio gives the **maximum** number of relationship instances an entity can participate in.

Ratio	Meaning	Example
1:1	One entity of A relates to at most one of B and vice versa	EMPLOYEE MANAGES DEPARTMENT
1:N	One A relates to many B, but each B relates to one A	DEPARTMENT has many EMPLOYEEs
N:1	Many A relate to one B	Many EMPLOYEEs work for one DEPARTMENT
M:N	Many A relate to many B	STUDENT enrolls in many COURSEs

Participation constraint

The participation constraint gives the **minimum** — whether every entity must take part. **Total participation** (double line) means compulsory; **partial participation** (single line) means optional. In the COMPANY diagram, every EMPLOYEE must belong to a department (total on the EMPLOYEE side of WORKS_FOR), but only one employee manages a department (partial on the DEPARTMENT side of MANAGES).

COMPANY Database — ER Diagram



WORKS_ON carries the descriptive attribute Hours. DEPENDENT is a weak entity (partial key Dependent_name).

Reading the diagram

Rectangles are entity sets, diamonds are relationship sets, and the labels on the connecting lines give the cardinality. A double line would mark total participation. Together, cardinality ratio plus participation constraint completely describe the structural constraint of a relationship, and both are needed before the diagram can be mapped to tables.

Q10. Explain the users of a DBMS and the responsibilities of a Database Administrator.

PYQ R25 Q1(c), B26 Q1(b), R24 model questions

Users of a DBMS

User	Role
Naive end users	Invoke pre-written application programs through forms and reports
Application programmers	Write programs in a host language with embedded DML
Sophisticated users	Analysts, scientists and engineers who write their own queries
DBA	Controls the schema, storage, authorisation, performance, backup and recovery
Database designers	Identify data, choose structures and define views

Responsibilities of the DBA

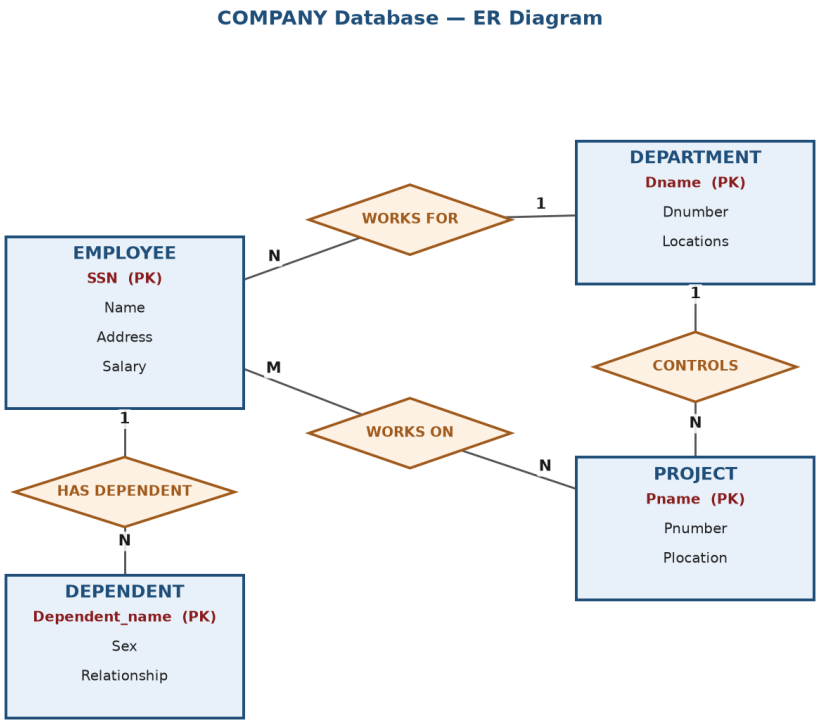
Area	Duties
Schema definition	Define the conceptual, external and internal schemas and their mappings
Storage and access	Choose file organisations, indexes and access paths; reorganise as needed
Authorisation	Grant and revoke privileges; define views to restrict access
Integrity	Declare and monitor constraints; ensure consistency across the database
Performance	Monitor usage, tune queries, plan capacity, resolve contention
Backup and recovery	Plan backups, maintain logs, restore after failure
User liaison	Identify requirements, choose the DBMS, train and support users
Documentation	Maintain the data dictionary and all system documentation

Q11. Design an ER diagram for a company database (departments, projects, employees, dependents).

PYQ R24 Q3(c) — the standard COMPANY schema

Problem

A company is organised into departments. Each department has a name, a number and a manager (with a start date), may have several locations, and controls several projects. Each project has a unique name, a unique number and one location. Each employee has name, SSN, address, salary, sex and birthdate, works for exactly one department, and may work on several projects with hours per week recorded. Employees may have dependents, each with name, sex, birthdate and relationship.



WORKS_ON carries the descriptive attribute Hours. DEPENDENT is a weak entity (partial key Dependent_name).

Design decisions

DEPARTMENT has a multivalued attribute Locations, which will become a separate table on mapping. MANAGES is a 1:1 relationship carrying the descriptive attribute Start_date. WORKS_FOR is N:1 with total participation on the EMPLOYEE side, since every employee must belong to a department. CONTROLS is 1:N from DEPARTMENT to PROJECT. WORKS_ON is M:N and carries Hours. DEPENDENT is a weak entity of EMPLOYEE through the identifying relationship HAS_DEPENDENT, with Dependent_name as partial key.

Mapping to tables

Table	Attributes	Key
EMPLOYEE	SSN, Name, Address, Salary, Sex, Birthdate, Dnumber	PK: SSN; FK: Dnumber to DEPARTMENT
DEPARTMENT	Dname, Dnumber, Mgr_ssn, Mgr_start_date	PK: Dnumber; FK: Mgr_ssn to EMPLOYEE
DEPT_LOC	Dnumber, Dlocation	PK: (Dnumber, Dlocation)
PROJECT	Pname, Pnumber, Plocation, Dnum	PK: Pnumber; FK: Dnum to DEPARTMENT
WORKS_ON	Essn, Pno, Hours	PK: (Essn, Pno); FKs to EMPLOYEE and PROJECT
DEPENDENT	Essn, Dependent_name, Sex, Birthdate, Relationship	PK: (Essn, Dependent_name); FK: Essn to EMPLOYEE

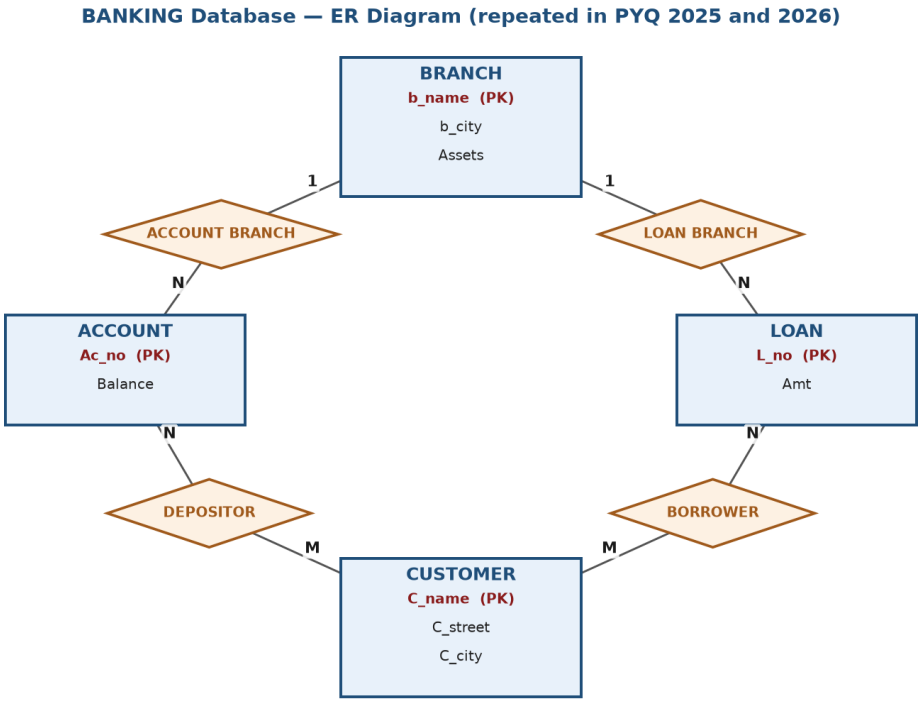
Q12. Design an ER diagram for a banking database and map it to relational tables.

PYQ B26 Q3(c), R24 Q3(c) — Figure 1 of both papers

Problem

A bank has many branches. Each branch is identified by branch name and has a city and assets. Each branch has multiple customers. A customer has a name, street and city and is given a unique

id by the branch. Customers can open Savings and Checking accounts. Branches provide loans, identified by loan number with an amount.



Maps to 6 tables: Account, Branch, Loan, Borrower, Customer, Depositor.

Design decisions

BRANCH is the owner of both ACCOUNT and LOAN through 1:N relationships, because each account and each loan belongs to exactly one branch. DEPOSITOR and BORROWER are M:N relationships between CUSTOMER and ACCOUNT, and between CUSTOMER and LOAN, because an account may be jointly held and a loan may have co-borrowers.

Mapping to tables (6 tables)

Table	Attributes	Key
Branch	b_name, b_city, Assets	PK: b_name
Account	Ac_no, Balance, b_name	PK: Ac_no; FK: b_name to Branch
Loan	L_no, Amt, b_name	PK: L_no; FK: b_name to Branch
Customer	C_name, C_street, C_city	PK: C_name
Depositor	C_name, Ac_no	PK: (C_name, Ac_no); both FKs
Borrower	C_name, L_no	PK: (C_name, L_no); both FKs

Note

The two 1:N relationships are merged into the many side, so b_name appears inside Account and Loan. The two M:N relationships each become a separate table. If Savings and Checking are treated as subclasses of Account, the specialization is mapped either as one table with a type attribute or as separate tables sharing Ac_no.

Q13. Compare the ER model with the EER model and explain the EER features.

PYQ R24 Q1(d), Q1(e)

ER Model	EER Model
Entity sets, attributes, relationships, keys	All of these plus subclasses, superclasses and inheritance
No support for class hierarchy	Supports generalization and specialization hierarchies
Cannot express union of entity sets	Supports category (union) types
No constraint on subclasses	Supports disjoint/overlapping and total/partial constraints
Relationships cannot be aggregated	Supports aggregation
Simpler to draw and teach	More expressive, closer to object-oriented design

The EER features

Subclass and superclass with attribute inheritance; **generalization** (bottom-up) and **specialization** (top-down); **category or union type**, where a subclass is a subset of the union of several superclasses; **aggregation**, treating a relationship as a higher-level entity; and the **constraints** on specialization — disjoint or overlapping, total or partial, and attribute-defined or user-defined.

Q14. Explain the anomalies of the hierarchical and network models.

PYQ R25 model questions

Insertion anomaly

A child record cannot be inserted without its parent. In a hierarchical model, a new employee cannot be stored until the department exists, and a new course cannot be recorded until some student enrolls in it.

Deletion anomaly

Deleting a parent deletes all its children automatically. Deleting the last employee of a department can erase the department's own details, because the parent record exists only through its children.

Update anomaly

If the same fact is repeated at several places in the tree or graph, changing it requires updating every copy; otherwise the database becomes inconsistent.

Structural rigidity

The hierarchical model cannot represent M:N relationships directly; they must be duplicated or converted into redundant trees. The network model avoids this but requires the programmer to know the pointer structure, so any change in the set structure forces the programs to be rewritten.

Why the relational model avoids these

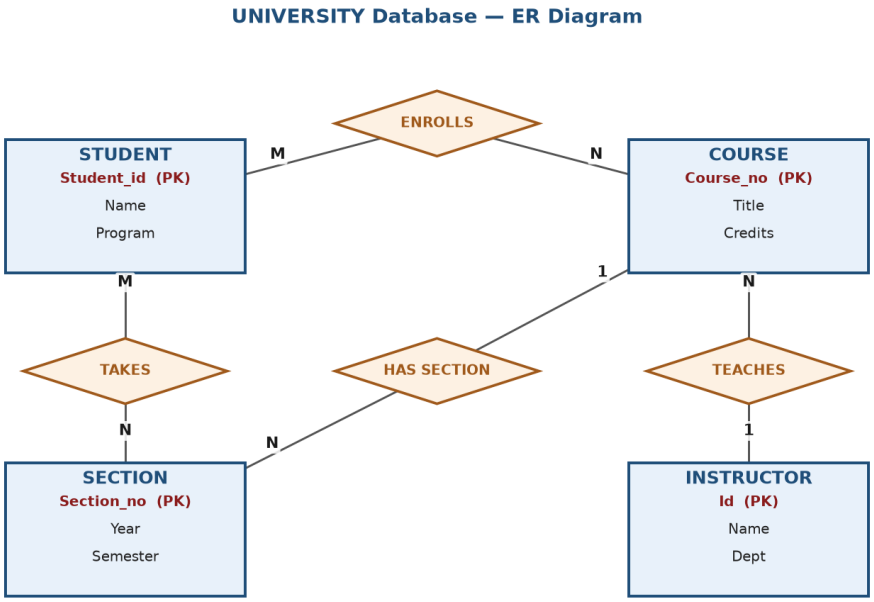
In the relational model, every fact is stored in exactly one relation determined by its functional dependencies, relationships are expressed by values rather than pointers, and normalization removes the repetition that causes the anomalies.

Q15. Design an ER diagram for a university registrar's office and map it to tables.

Model question — Unit 1

Problem

The registrar maintains courses (number, title, credits, syllabus, prerequisites), course offerings (course number, year, semester, section number, instructors, timings, classroom), students (student-id, name, program) and instructors (identification number, name, department, title). The enrolment of students in courses and the grades awarded must be modelled.



SECTION is a weak entity of COURSE; its partial key is Section_no. Grade is a descriptive attribute of TAKES.

Entities and keys

STUDENT with key Student_id, COURSE with key Course_no, INSTRUCTOR with key Id, and SECTION — the course offering — which is a **weak entity** of COURSE with partial key (Year, Semester, Section_no).

Relationships

HAS_SECTION is 1:N from COURSE to SECTION with total participation on the SECTION side, because an offering cannot exist without its course. TEACHES is M:N between INSTRUCTOR and SECTION, because one offering may have several instructors. TAKES is M:N between STUDENT and SECTION and carries the descriptive attribute **Grade**. PREREQ is a **recursive M:N** relationship on COURSE.

Mapping to tables

Table	Attributes	Key and constraints
STUDENT	Student_id, Name, Program	PK: Student_id
INSTRUCTOR	Id, Name, Dept, Title	PK: Id
COURSE	Course_no, Title, Credits, Syllabus	PK: Course_no
PREREQ	Course_no, Pre_course_no	PK: (Course_no, Pre_course_no); both FKs
SECTION	Course_no, Year, Semester, Section_no, Classroom, Timing	PK: (Course_no, Year, Semester, Section_no); FK: Course_no
TEACHES	Id, Course_no, Year, Semester, Section_no	PK: all four columns; FKs
TAKES	Student_id, Course_no, Year, Semester, Section_no, Grade	PK: all five columns; FKs

The two M:N relationships become their own tables, the 1:N relationship is merged into the many side, and the weak entity receives the owner's key as part of its primary key.

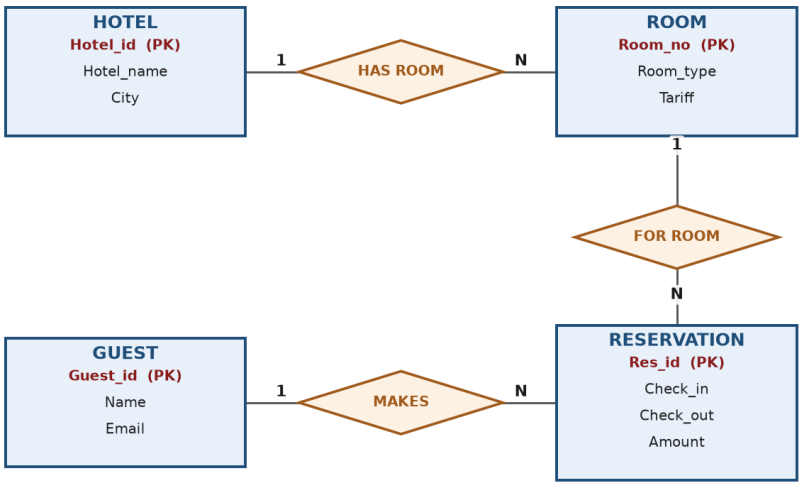
Q16. Design an ER diagram for a hotel management system and map it to tables.

Model question — Unit 1

Problem

A hotel chain keeps Hotels (Hotel_ID, Hotel_Name, Address, City, Phone), Rooms (Room_No, Room_Type, Tariff, Status), Guests (Guest_ID, Name, Address, Phone, Email) and Employees (Employee_ID, Name, Designation, Salary). Guests reserve rooms, and every reservation records Check_In_Date, Check_Out_Date, Number_of_Guests and Total_Amount. Services such as Laundry, Wi-Fi, Restaurant and Spa have Service_ID, Service_Name and Charge, and a room may receive many services.

HOTEL Database — ER Diagram



Reservation is an associative entity that resolves the M:N between GUEST and ROOM.

Design decisions

HAS_ROOM is 1:N from HOTEL to ROOM with total participation on the ROOM side, because every room belongs to exactly one hotel. RESERVATION is an **associative entity** that resolves the M:N between GUEST and ROOM and carries its own attributes. WORKS_IN is 1:N from HOTEL to EMPLOYEE. PROVIDES is M:N between ROOM and SERVICE.

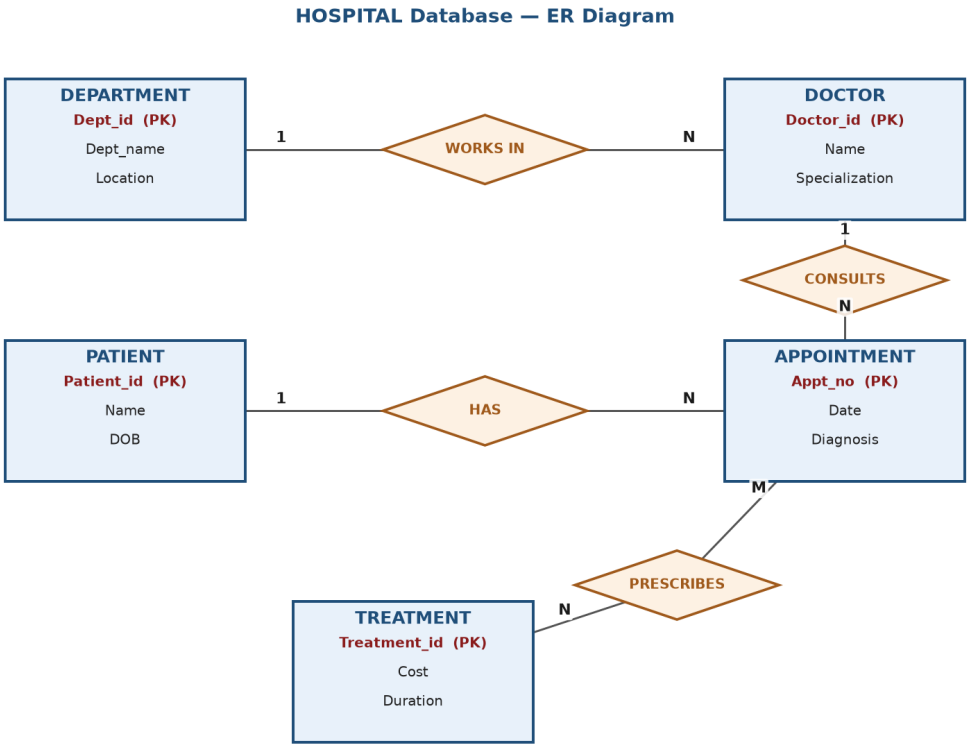
Mapping

Table	Attributes	Key and constraints
HOTEL	Hotel_ID, Hotel_Name, Address, City, Phone	PK: Hotel_ID
ROOM	Hotel_ID, Room_No, Room_Type, Tariff, Status	PK: (Hotel_ID, Room_No); FK: Hotel_ID
GUEST	Guest_ID, Name, Address, Phone, Email	PK: Guest_ID
EMPLOYEE	Employee_ID, Name, Designation, Salary, Hotel_ID	PK: Employee_ID; FK: Hotel_ID
RESERVATION	Res_ID, Guest_ID, Hotel_ID, Room_No, Check_In, Check_Out, No_of_Guests, Amount	PK: Res_ID; FKs to GUEST and ROOM
SERVICE	Service_ID, Service_Name, Charge	PK: Service_ID
ROOM_SERVICE	Hotel_ID, Room_No, Service_ID, Service_Date	PK: all four columns; FKs

Room_No alone is not unique across the chain, so the primary key of ROOM is composite — the hotel plus the room number.

Q17. Convert the hospital ER model into relational tables.

Model question — Unit 1



Appointment is modelled as an associative entity because it carries its own attributes.

The model

DEPARTMENT (Dept_id, Dept_name, Location) employs DOCTOR (Doctor_id, Name, Specialization) through WORKS_IN, which is 1:N with total participation on the doctor side. PATIENT (Patient_id, Name, DOB) makes an APPOINTMENT (Appt_no, Date, Diagnosis) through HAS, which is 1:N. APPOINTMENT and TREATMENT (Treatment_id, Cost, Duration) are related through PRESCRIBES, which is M:N.

Table	Attributes	Key and constraints
DEPARTMENT	Dept_id, Dept_name, Location	PK: Dept_id
DOCTOR	Doctor_id, Name, Specialization, Dept_id	PK: Doctor_id; FK: Dept_id
PATIENT	Patient_id, Name, DOB	PK: Patient_id
APPOINTMENT	Appt_no, Patient_id, Doctor_id, Date, Diagnosis	PK: Appt_no; FKs to PATIENT and DOCTOR
TREATMENT	Treatment_id, Treatment_name, Cost, Duration	PK: Treatment_id
PRESCRIBES	Appt_no, Treatment_id, Dosage	PK: (Appt_no, Treatment_id); both FKs

The 1:N relationships are merged into the many side, so Dept_id appears inside DOCTOR and Patient_id and Doctor_id inside APPOINTMENT. The M:N relationship PRESCRIBES becomes its own table and carries Dosage as a descriptive attribute.

Q18. Explain data integrity and the various integrity constraints.

Model question — Unit 1

What data integrity means

Data integrity is the guarantee that stored data is accurate, consistent and valid — that it satisfies every rule the organisation has declared. The DBMS enforces it on every insert, update and delete, so consistency does not depend on each application program being correct.

Constraint	Rule	Example
Domain	Each value must come from the attribute's domain	Age must be an integer between 0 and 120
Key	No two tuples may share a primary key value	Two students cannot both have SID = S01
Entity integrity	The primary key may not be NULL	SID can never be empty
Referential integrity	A foreign key must match an existing key or be NULL	DeptID D9 is rejected if DEPARTMENT has no D9
Semantic	Business rules the schema cannot express	Salary may not exceed the manager's salary

Referential actions

When a parent row is deleted or its key is updated, the DBMS may **RESTRICT** the operation, **CASCADE** it to the children, **SET NULL** in the children, or **SET DEFAULT**.

Why it matters

Each rule is written once, stored in the data dictionary, and enforced by the DBMS for every user and every program. Without declared constraints the same rule would have to be coded into every application, and one careless program would be enough to corrupt the data.