Database Management Systems

Data models and the DBMS

The notes for the lesson “Data models and the DBMS”: chapters 1.3 to 1.4, with their figures, worked examples, exam traps, quizzes and exam question bank, in one file.

2 chapters | Chapters 1.3 to 1.4

About these notes

These are the notes for the University free n easy video lesson **Data models and the DBMS** on Database Management Systems: chapters 1.3 to 1.4, with every figure, worked example, exam trap and quiz, and the long-answer exam questions on these chapters.

1.3 Data models: hierarchical, network, relational, object-oriented

A **data model** is a set of concepts for describing a database: the **structure** of the data (types and relationships), the **operations** allowed on it, and the **constraints** it must obey. Every DBMS is built on one data model, and the model decides how relationships are stored and how queries are written. Four models matter for the exam: the hierarchical and network models of the 1960s and 1970s, which link records with pointers; the relational model, which links rows by matching values and runs most databases today; and the object-oriented model, which stores programming-language objects directly.

Kinds of data models

- **Conceptual (high-level) models** describe data as users think of it: entities, attributes and relationships. The ER model of Chapters 1.5 to 1.8 is the main one.
- **Representational (implementation) models** are what commercial DBMSs implement: hierarchical, network, relational, and object-oriented or object-relational.
- **Physical (low-level) models** describe storage: record formats, ordering and access paths (a later lesson).
- **Semi-structured models** such as XML and JSON let each item carry its own tags, so items of the same kind may have different fields.

Running example for all four: the CSE department offers DBMS and OS; Asha takes both, Ravi takes DBMS. Student–course is a many-to-many (M:N) relationship, and the way each model handles M:N is the key difference.

Hierarchical and network models

The **hierarchical model** arranges records in trees. Each parent–child link is one-to-many (1:N): a parent may have many children, but every child has exactly one parent. The root is the entry point, and a program reaches data by walking down from the root. IBM's IMS, built in the late 1960s for the Apollo space programme, is the classic example and still runs in some banks. Because a child has one parent, Asha must be stored twice to sit under both DBMS and OS (Figure 1.3.1a). A child cannot exist without its parent, and deleting a parent deletes its subtree.

The **network model** allows a record to belong to several parent–child links, so the records form a graph. Each link type is a **set type** with one **owner** record type and one **member** record type, still 1:N. An M:N relationship is stored through a **link record** (here, one enrol record per student–course pair) that is a member of two sets (Figure 1.3.1b). The model was standardised by the CODASYL Data Base Task Group (DBTG) in 1971, building on Charles Bachman's IDS; IDMS is a well-known product. Programs navigate one record at a time along pointers, with commands such as FIND and GET, so the code depends on the access path.

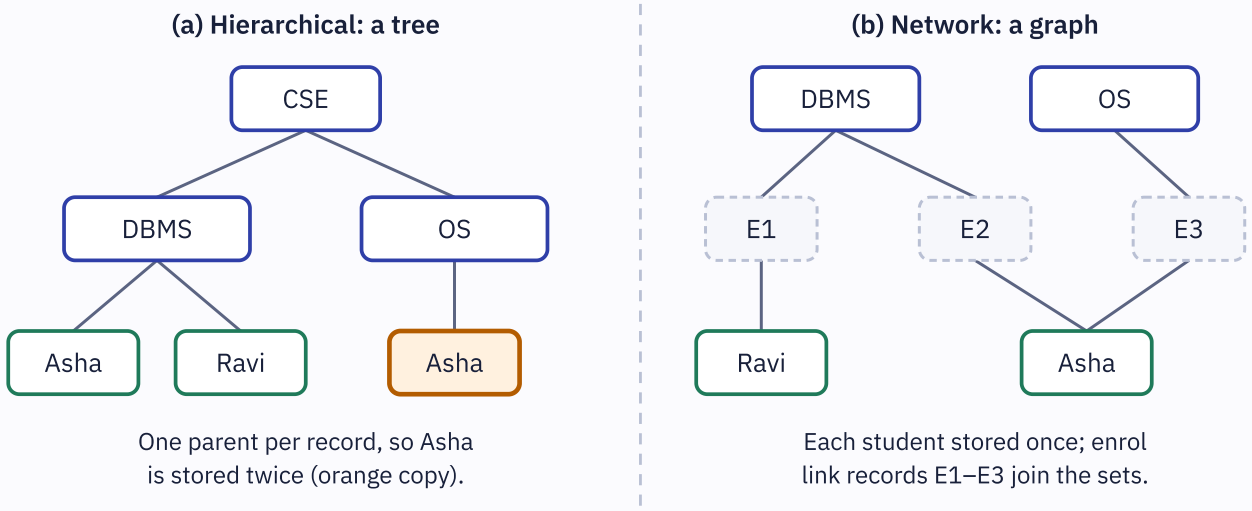

Figure 1.3.1 The same enrolments in the two pointer-based models. Notice that the tree must repeat Asha, while the network stores her once and uses a link record (dashed) for each student–course pair.

Relational and object-oriented models

The **relational model**, proposed by E. F. Codd of IBM in 1970, stores all data in tables (relations). There are no stored pointers between tables: a row of ENROL refers to a student by holding the

student's roll number, and the DBMS matches equal values when asked (Figure 1.3.2c). Queries are declarative and work on whole sets of rows ("give me all courses of roll 2"), and the DBMS chooses the access path. This brings strong data independence, and it is why Oracle, MySQL, PostgreSQL, SQL Server and Db2 are all relational. Later lessons are about this model.

The **object-oriented model** stores objects as a program sees them: each object has a system-given **object identifier (OID)**, attributes and methods; objects belong to classes with inheritance and encapsulation; and an object refers to another directly by its OID (Figure 1.3.2d). Nested and complex objects, such as a machine made of parts made of sub-parts, need no flattening into tables. The ODMG standard defined ODL (definitions) and OQL (queries). It suits CAD, engineering and multimedia data. The **object-relational model** keeps tables and SQL but adds user-defined types, inheritance and arrays, as in PostgreSQL and Oracle.

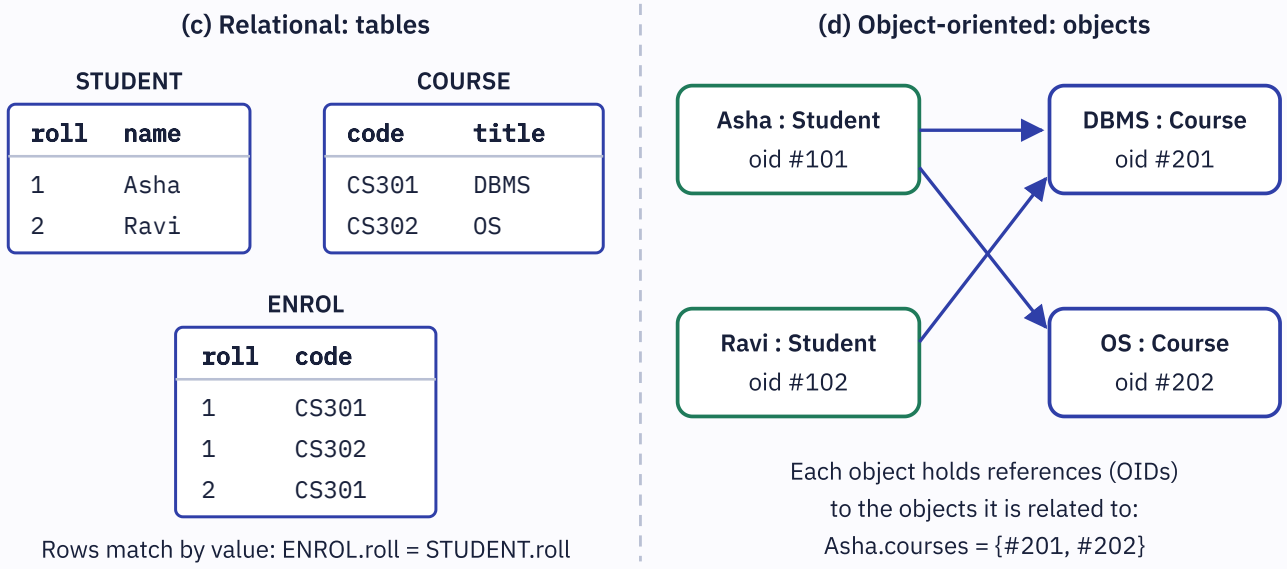

Figure 1.3.2 The same enrolments as tables and as objects. Notice that the relational model needs a third table, ENROL, whose rows pair values, while the objects point to each other directly by identifier.

MODEL	STRUCTURE	RELATIONSHIPS BY	ACCESS	EXAMPLE
Hierarchical	Trees of records	Parent–child pointers (1:N only)	Navigate from root	IBM IMS
Network	Graph of records	Owner–member sets; link records for M:N	Navigate, record at a time	IDMS (CODASYL)
Relational	Tables of rows	Matching values (foreign keys)	Declarative, set at a time (SQL)	PostgreSQL, Oracle

MODEL	STRUCTURE	RELATIONSHIPS BY	ACCESS	EXAMPLE
Object-oriented	Objects in classes	Object identifiers	Methods, path expressions, OQL	ODMG-style systems

WORKED EXAMPLE

Question. A department has 3 courses and 50 students, with 120 enrolments in all. A student record is 100 bytes and a course record 60 bytes. A network link record holds two 8-byte pointers; a relational ENROL row holds a 4-byte roll and an 8-byte course code. Find the bytes used under (a) the hierarchical model with course as parent, (b) the network model, (c) the relational model. (d) How many records change when one student who takes 3 courses changes phone number? (6 marks)

1. (a) Course records: 3 × 60 = 180. Each enrolment puts a full student record under a course: 120 × 100 = 12,000. Total = **12,180 bytes**.
2. (b) Courses 180; students stored once: 50 × 100 = 5,000; link records 120 × 16 = 1,920. Total = **7,100 bytes**.
3. (c) COURSE 180; STUDENT 5,000; ENROL 120 × (4 + 8) = 1,440. Total = **6,620 bytes**.
4. (d) Hierarchical: the student has 3 copies, so **3 records** change, and missing one leaves an inconsistency. Network and relational: **1 record**.
5. Conclusion: the tree spends 12,180 − 6,620 = 5,560 extra bytes (84% more than relational) on repeated students, and adds update risk. Both effects grow with every M:N pair.

EXAM TRAPS

- **Hierarchical = one parent per child.** A parent may have many children; it is the child that has one parent. So M:N needs duplicated records (or IMS's virtual pointer records).
- **Network sets are still 1:N.** The graph appears because a record type can be a member of several set types; M:N always goes through a link record.
- **Relational tables are linked by values, not pointers.** Writing "the relational model uses pointers between tables" loses the mark.
- **ER is a conceptual model, not a DBMS product.** No database "runs" the ER model; designs drawn in ER are mapped to the relational model (Chapter 1.9).
- Object-oriented and object-relational are different: the second keeps tables and SQL.

CHECK YOURSELF

- REMEMBER **What three things does a data model specify?** 2 marks

  The structure of the data (types, relationships), the operations allowed on it, and the constraints it must satisfy.

- REMEMBER **Who proposed the relational model, and in which year?** 1 mark

  E. F. Codd of IBM, in 1970.

- UNDERSTAND **Why can the hierarchical model not represent an M:N relationship without redundancy?** 3 marks

  Every child record has exactly one parent. A student in two courses needs two parents, so the student record must be copied under each course.

- UNDERSTAND **Contrast record-at-a-time navigation with set-at-a-time queries.** 4 marks

  Navigational (network, hierarchical): the program follows pointers one record at a time, so it fixes the access path and breaks if the structure changes. Set-at-a-time (relational SQL): the query states which rows are wanted and the DBMS chooses the path, giving data independence.

- APPLY **4 courses, 30 students, 75 enrolments, 80-byte student records. How many bytes of student records in the hierarchical model (course as parent) and in the relational model?** 3 marks

  Hierarchical: 75 × 80 = **6,000 bytes**. Relational: 30 × 80 = **2,400 bytes** (plus the small ENROL table). Duplication factor 75 ÷ 30 = 2.5.

- APPLY **Teachers teach many subjects and subjects have many teachers. Describe the network schema.** 3 marks

  Record types TEACHER and SUBJECT, plus a link record type TEACHES. Two set types: TEACHER (owner) → TEACHES (member) and SUBJECT (owner) → TEACHES (member). One TEACHES record per teacher–subject pair.

– ANALYZE **A design firm stores machines made of assemblies made of parts, each with drawings and methods to compute weight. Which model fits best? Justify.** 5 marks

Object-oriented (or object-relational): nested, complex objects with behaviour are stored as they are, references replace many joins, and inheritance models part families. A pure relational design needs many tables and recursive joins; a hierarchy breaks when a part is shared by two assemblies.

BEYOND THE SYLLABUS

This is not for the exam. The old models came back under new names. Document databases such as MongoDB store JSON trees, hierarchical in spirit, and face the same M:N duplication choice. Graph databases such as Neo4j store nodes and edges like the network model but add declarative queries; ISO published GQL, a standard graph query language, in 2024. A later lesson returns to these NoSQL systems. Search next: "document model versus relational model" and "property graph model".

1.4 Database languages, users and DBMS architecture

People talk to a DBMS in database languages: one part defines the schema, another reads and changes data, others control access and transactions. Different people use it in different ways, from a passenger booking a ticket to the administrator tuning the disks. Inside, the DBMS has two main parts, a **query processor** that turns requests into actions and a **storage manager** that moves data between disk and memory safely.

Database languages

- **Data definition language (DDL)** defines and changes the schema: `CREATE`, `ALTER`, `DROP`, `TRUNCATE`. Its output is metadata (data about data), stored in the **data dictionary** or system catalogue. Elmasri and Navathe also name a storage definition language (SDL) for the internal schema and a view definition language (VDL) for external schemas; in SQL all three are part of one language.
- **Data manipulation language (DML)** retrieves, inserts, deletes and updates data: `SELECT`, `INSERT`, `UPDATE`, `DELETE`. A **procedural** DML makes the user say what data is needed and how to get it, step by step (network-model FIND and GET, relational algebra). A **declarative**

(non-procedural) DML asks only for what is needed (SQL). The retrieval part of a DML is called a **query language**.

- **Data control language (DCL)** gives and takes away rights: `GRANT`, `REVOKE`.
- **Transaction control language (TCL)**: `COMMIT`, `ROLLBACK`, `SAVEPOINT` (a later lesson).

The SQL below (run in PostgreSQL, with a role `exam_cell` already created) uses all four; the last lines show the query's output.

```
BEGIN;                                                    -- TCL
CREATE TABLE student (roll INT PRIMARY KEY, name VARCHAR(40),
       cgpa NUMERIC(4,2) CHECK (cgpa BETWEEN 0 AND 10));      -- DDL
INSERT INTO student VALUES (101, 'Asha', 8.70), (102, 'Ravi', 7.95);   -- DML
GRANT SELECT ON student TO exam_cell;                         -- DCL
COMMIT;                                                   -- TCL
SELECT name FROM student WHERE cgpa > 8;                     -- DML query

 name
------
 Asha
(1 row)
```

Database users and the administrator

- **Naive (parametric) users** use ready-made screens: an IRCTC passenger, a bank teller, a person at an ATM. They never see SQL.
- **Application programmers** write those screens and programs in Java, Python or C#, sending SQL through interfaces such as JDBC.
- **Sophisticated users** (analysts) write their own queries or use reporting and analysis tools. Elmasri and Navathe also name **casual users**, who query now and then, and **stand-alone users** with personal databases.
- **Database designers** decide what data to store and draw the ER diagram (Chapters 1.5 to 1.9).
- The **database administrator (DBA)** has central control: defines the schema and storage structures, changes them as needs grow, grants and revokes access, takes backups, watches disk space and tunes performance.

Components of a DBMS

Figure 1.4.1 shows the parts and the order in which a request passes through them.

1. **DDL interpreter**: reads DDL statements and records the definitions in the data dictionary.
2. **DML compiler**: parses a query and translates it into an evaluation plan of low-level steps. Because one query has many equivalent plans, it performs **query optimisation**, choosing the cheapest plan using statistics about the data (a later lesson).

3. **Query evaluation engine**: executes the chosen plan.
4. **Authorization and integrity manager**: checks the user's rights and the integrity constraints.
5. **Transaction manager**: keeps the database consistent despite failures (recovery manager) and many users at once (concurrency-control manager) (a later lesson).
6. **File manager**: allocates disk space and manages the files that hold the data.
7. **Buffer manager**: brings disk blocks into main memory and decides which to keep, so that most requests are served without a slow disk read.

On disk live the **data files**, the **data dictionary** (metadata), the **indices** that find rows quickly, and **statistical data** used by the optimiser.

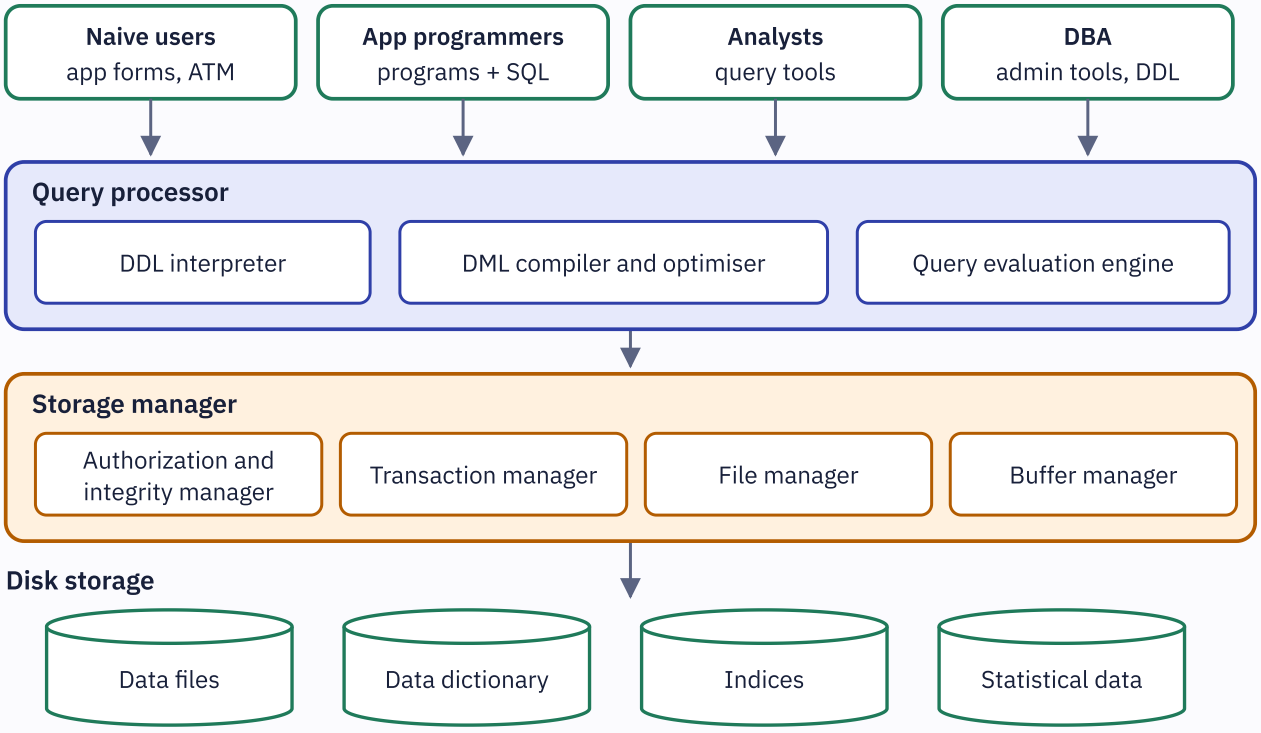

Figure 1.4.1 The structure of a DBMS. Requests flow down: users and their tools, then the query processor, then the storage manager, then disk. Notice that the DBA's DDL ends in the data dictionary, and that the optimiser's statistics also live on disk.

Where the DBMS runs: tiers and types

In a **centralised** system the DBMS and its users share one computer. In **client–server** systems the DBMS runs on a server. In a **two-tier** design the client program holds the screens and the business rules and talks to the database server directly. In a **three-tier** design the client (often a browser or mobile app) talks only to an **application server**, which holds the business rules and talks to the database (Figure 1.4.2). Three tiers are safer, because clients never touch the

database, and they scale, because a few application servers share a small pool of database connections.

Database systems are also classed by number of users (single-user or multi-user), number of sites (centralised or distributed, a later lesson), data model (Chapter 1.3) and purpose (transaction processing for daily operations, or analytical data warehouses).

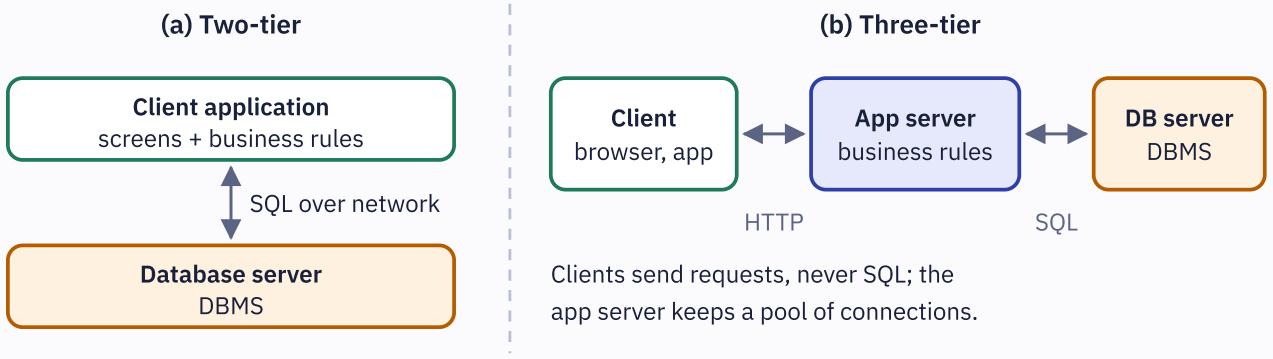

Figure 1.4.2 Two-tier and three-tier client–server architectures. Notice where the business rules live: in every client in (a), once in the application server in (b).

WORKED EXAMPLE

Question. A university results portal has 20,000 students online at the peak. Each open database connection uses 5 MB of memory on the database server. (a) In a two-tier design every client holds one connection. How much server memory do connections need? (b) In a three-tier design, 8 application servers each keep a pool of 25 connections. Repeat. (c) Classify each statement and say who issues it: `CREATE TABLE result(...)`; `INSERT INTO result ...`; `SELECT * FROM result WHERE roll = 2117`; `GRANT SELECT ON result TO portal_app`; `COMMIT`. (d) Trace the SELECT through the components of Figure 1.4.1. (8 marks)

1. (a) Connections = 20,000. Memory = 20,000 × 5 MB = **1,00,000 MB**, about 98 GB, just for connections.
2. (b) Connections = 8 × 25 = 200. Memory = 200 × 5 MB = **1,000 MB**, about 1 GB: one hundredth of (a). Requests queue briefly for a free pooled connection instead of each holding one.
3. (c) CREATE TABLE: DDL, by the DBA. INSERT: DML, by the exam cell's application program. SELECT: DML query, sent by the portal on behalf of a naive user (the student). GRANT: DCL, by the DBA. COMMIT: TCL, by the application program after the upload.
4. (d) The app server sends the SELECT. The DML compiler parses it, checks names in the data dictionary, and the optimiser uses statistics to choose the index on roll over a full scan. The authorization manager confirms `portal_app` may read `result`. The evaluation engine runs the plan, asking the buffer manager for the index and data blocks; blocks not in memory are read through the file manager. The row goes back to the app server, which formats the page.

EXAM TRAPS

- **DDL output is metadata.** It goes into the data dictionary, not into the data files.
- **SQL is declarative; relational algebra and PL/SQL are procedural.** Do not call SQL procedural because it has many steps in a long query.
- **The DML compiler optimises; the evaluation engine executes.** Students often give optimisation to the engine.
- **The transaction manager has two halves:** concurrency control and recovery. Name both.
- **Three-tier is about machines, three-schema about levels of data description.** A question on one does not want the other.

CHECK YOURSELF

– REMEMBER **Expand DDL, DML, DCL and TCL and give two commands of each.** 4 marks

Data definition (CREATE, ALTER); data manipulation (SELECT, UPDATE); data control (GRANT, REVOKE); transaction control (COMMIT, ROLLBACK).

– REMEMBER **List five duties of a DBA.** 3 marks

Schema definition; storage structure and access-method definition; schema and physical changes; granting and revoking authorisation; routine maintenance: backups, disk space, performance monitoring.

– UNDERSTAND **Distinguish procedural and declarative DMLs with an example of each.** 4 marks

Procedural: the user states what data and how to get it, step by step (network FIND/GET, relational algebra). Declarative: only what is wanted (SQL SELECT ... WHERE ...); the DBMS works out how.

– UNDERSTAND **Why does the buffer manager matter for performance?** 3 marks

Disk reads are thousands of times slower than memory reads. By keeping frequently used blocks in main memory and choosing which to evict, the buffer manager lets most requests finish without touching the disk.

– APPLY **Classify: ALTER TABLE, UPDATE, REVOKE, ROLLBACK, TRUNCATE, SELECT.** 3 marks

ALTER: DDL. UPDATE: DML. REVOKE: DCL. ROLLBACK: TCL. TRUNCATE: DDL (in most systems). SELECT: DML (query).

– APPLY **6,000 clients connect two-tier; three-tier uses 5 app servers with 40 pooled connections each. At 4 MB per connection, compare server memory.** 3 marks

Two-tier: 6,000 × 4 = **24,000 MB**. Three-tier: 5 × 40 = 200 connections × 4 = **800 MB**, 30 times less.

– ANALYZE **Argue why a railway booking system should be three-tier rather than two-tier.** 5 marks

Millions of clients cannot each hold a database connection; pooling in app servers caps it. Fare and quota rules change often and must change in one place, not in every phone app. Clients that never send SQL cannot attack the database directly. App servers can be added to scale, while the database stays one consistent store.

BEYOND THE SYLLABUS

This is not for the exam. Many organisations now rent a managed database in the cloud (Amazon RDS, Google Cloud SQL, Azure SQL Database), where the provider plays much of the DBA's role. Some cloud designs split the storage manager from the query processor, so many compute nodes share one storage layer. Embedded engines such as SQLite and DuckDB have no server at all: the DBMS is a library inside the program. Search next: "connection pooling PgBouncer" and "separation of storage and compute".

Key terms of this lesson

TERM	MEANING OR FORMULA	CHAPTER
Data models	Hierarchical (tree, 1 parent), network (graph, owner–member sets), relational (tables, values), object-oriented (objects, OIDs)	1.3
Languages	DDL (CREATE, ALTER, DROP), DML (SELECT, INSERT, UPDATE, DELETE), DCL (GRANT, REVOKE), TCL (COMMIT, ROLLBACK)	1.4
DBMS components	Query processor (DDL interpreter, DML compiler, evaluation engine) + storage manager (authorization, transaction, file, buffer managers)	1.4

Exam question bank

Long-answer exam questions on these chapters, each with an outline of the points an examiner looks for. Answer one in full on paper before you read its outline.

Remember and understand

- UNDERSTAND **Q3. Compare the hierarchical, network, relational and object-oriented data models, with a diagram of the same data in each.** 13 marks

  Definition of a data model (structure, operations, constraints) and its categories. Figures 1.3.1 and 1.3.2 for one M:N example. Comparison table: structure, how relationships are stored, access style, example system, weakness. Codd 1970; IMS; CODASYL DBTG; ODMG; object-relational.

- REMEMBER **Q4. Draw the overall structure of a DBMS and explain each component. Classify database users and list the duties of the DBA.** 13 marks

  Figure 1.4.1: users and tools, query processor (DDL interpreter, DML compiler and optimiser, evaluation engine), storage manager (authorization and integrity, transaction, file, buffer managers), disk storage (data, dictionary, indices, statistics). Naive, application programmers, sophisticated, casual users; DBA duties. DDL, DML, DCL, TCL with examples.

Analyze and evaluate

- ANALYZE **Q11. A food-delivery start-up keeps orders in spreadsheets shared by 30 staff in 3 cities. Evaluate moving to a DBMS: which file-system problems will it meet, how the three-schema architecture helps as it grows, and what client–server architecture it should use.** 13 marks

  Spans 1.1, 1.2 and 1.4. Problems: redundancy (customer address per order), lost updates when two staff edit, no atomic order-plus-payment, no field-level security, slow new reports. Three levels: views for kitchen, riders and accounts; storage tuned without changing apps. Three-tier with app servers and pooled connections; numbers for connection memory. Costs of a DBMS.

Glossary (terms used in this lesson)

Attribute	A property of an entity or relationship, with values from a domain.
Data dictionary	The DBMS's store of metadata: table definitions, constraints, users and statistics.
Data independence	The ability to change the schema at one level without changing the next higher level or the programs.

Data model	A set of concepts for describing data structure, operations and constraints.
Database administrator (DBA)	The person with central control of the database: schema, storage, access rights, backups and tuning.
DBMS	Software that stores a database and lets many users define, query, update and protect it.
Foreign key	Columns of one table whose values must match the primary key of another table.
Primary key	The candidate key chosen as the main identifier of an entity set or table.
Schema	The overall design of the database, which changes rarely.

Sources and further reading

Textbooks

- Abraham Silberschatz, Henry F. Korth and S. Sudarshan, *Database System Concepts*, 7th ed., McGraw-Hill. Chapter 1, "Introduction" (purpose, views of data, languages, architecture, users), and the chapter "Database Design Using the E-R Model" (entity and relationship sets, weak entity sets, extended features, reduction to relational schemas).
- Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems*, 7th ed., Pearson. Chapters on databases and database users, database system concepts and architecture, the ER model, the enhanced ER model, and ER- and EER-to-relational mapping.
- Raghu Ramakrishnan and Johannes Gehrke, *Database Management Systems*, 3rd ed., McGraw-Hill: overview of database systems and introduction to database design with the ER model.
- C. J. Date, *An Introduction to Database Systems*, 8th ed., Pearson: the ANSI/SPARC three-level architecture.
- Peter P. Chen, "The Entity-Relationship Model: Toward a Unified View of Data", *ACM Transactions on Database Systems*, 1976.
- E. F. Codd, "A Relational Model of Data for Large Shared Data Banks", *Communications of the ACM*, 1970.

Web resources

The writing session's network could not open web pages, so these were not opened. The NPTEL course number, the CODASYL DBTG report date (1971) and the ISO GQL publication (2024) were checked through web-search summaries only.

- Slides and resources for Silberschatz, Korth and Sudarshan: db-book.com.
- NPTEL, "Data Base Management System" (Partha Pratim Das): nptel.ac.in/courses/106105175.
- PostgreSQL documentation, CREATE TABLE and constraints: postgresql.org/docs/current.

Syllabus: the University free n easy universal syllabus, built from top universities' syllabi.

Built for the University free n easy YouTube channel from the universal syllabus of Database Management Systems (Oct 2026). Written with top universities' syllabi and standard textbooks; check a formula against your prescribed book before an exam, and report mistakes on the channel.