DATABASE ENGINEERING

Unit 2

Question Bank — Relational Data Model and Mapping

Section A — 41 short-answer questions (2 marks each)

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

Section C — 14 long-answer questions (7 marks each), with diagrams

Merges the Unit II syllabus topics, the model question bank and every Unit II 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 relation, tuple, attribute and domain.

PYQ R25 Q1(d)

A **relation** is a table of values in which each row represents one entity and each column one property; formally it is a subset of the Cartesian product of a set of domains. A **tuple** is one row of the relation. An **attribute** is one named column. A **domain** is the set of atomic values that an attribute may take — for example the domain of Age is the set of integers from 0 to 120.

Q2. What is relation degree and relation cardinality?

PYQ R25 Q1(d)

The **degree** (or arity) of a relation is the number of attributes, that is the number of columns. The **cardinality** of a relation is the number of tuples, that is the number of rows. A relation with degree 4 and cardinality 12 has 4 columns and 12 rows.

Q3. What is a relation schema? What is a relation instance?

A **relation schema** is the name of the relation together with its list of attributes, written R(A1, A2, ..., An); it describes the structure and does not change with time. A **relation instance** is the actual set of tuples present in the relation at a particular moment; it changes whenever rows are inserted, deleted or updated.

Q4. Differentiate between schema and instance.

PYQ model questions, DBE_UNIT_2_2_1

Schema	Instance
Description of the database structure	Actual data present at a moment in time
Does not change frequently	Changes with every insert, delete or update
Also called intension	Also called extension or database state
Defined at design time using DDL	Populated and modified using DML
Example: STUDENT(SID, Name, Dept)	The current rows of STUDENT

Q5. Differentiate relation intension from relation extension.

PYQ model questions

Intension is the permanent, structural part of a relation — its name, attributes, domains and constraints; it is the schema. **Extension** is the changing part — the set of tuples currently stored. Intension is what the DBA designs; extension is what the users add and modify.

Q6. Define super key.

PYQ B26 Q1(a), R25 Q3(c)

A **super key** is a set of attributes whose values uniquely identify each tuple of the relation. It may contain extra attributes that are not needed for identification. Example: in STUDENT(SID, Name, Dept), both {SID} and {SID, Name} are super keys.

Q7. Define candidate key.

A **candidate key** is a **minimal super key** — a super key from which no attribute can be removed without losing the uniqueness property. Example: if SID and Email are each unique, then {SID} and {Email} are candidate keys while {SID, Email} is only a super key.

Q8. Define primary key and state its properties.

PYQ B26 Q1(a)

The **primary key** is the candidate key chosen by the designer to identify tuples of the relation. Its properties are: it is unique for every tuple; it can never be NULL (entity integrity); a relation can have only one primary key; and it is usually the smallest candidate key.

Q9. Define foreign key.

PYQ B26 Q1(a)

A **foreign key** is an attribute (or set of attributes) in one relation whose values must either match a value of the primary key of another relation or be entirely NULL. It is how relationships are represented in the relational model. Example: DeptID in EMPLOYEE is a foreign key referencing DEPARTMENT(DeptID).

Q10. Define alternate key.

An **alternate key** is a candidate key that was not chosen as the primary key. Example: if {SID} and {Email} are candidate keys and SID is chosen as the primary key, then Email is an alternate key.

Q11. Define composite key.

A **composite key** is a key made up of two or more attributes together. Example: in ORDER_ITEM(OrderID, ProductID, Quantity), neither OrderID nor ProductID alone is unique, but the pair (OrderID, ProductID) is, so it is a composite primary key.

Q12. "All primary keys are candidate keys." Justify.

PYQ model questions

A primary key is by definition one of the candidate keys, chosen by the designer from among them; therefore every primary key is a candidate key. The converse is not true: a relation may have several candidate keys but only one of them is designated primary, and the rest become alternate keys. Hence the primary key set is a subset of the candidate key set.

Q13. Can a table have more than one candidate key? Which becomes primary?

Yes. A table may have several candidate keys — for example STUDENT(SID, RollNo, Email) may have three. Any one of them is chosen as the primary key, usually the shortest and most stable; the remaining candidate keys become alternate keys.

Q14. Can a foreign key be NULL? Can it contain duplicate values?

A foreign key **can be NULL** when the relationship is optional — an employee who is not yet assigned to a project may have a NULL ProjectID. A foreign key **can contain duplicate values**, because many tuples may reference the same parent — many employees share the same DeptID. What it cannot contain is a value that does not exist in the referenced primary key.

Q15. Differentiate primary key from foreign key.

PYQ B26 Q1(a), model questions

Primary Key	Foreign Key
Uniquely identifies a tuple in its own relation	References the primary key of another relation
Cannot be NULL	May be NULL if the relationship is optional
Only one per relation	A relation may have many foreign keys
Enforces entity integrity	Enforces referential integrity

Primary Key	Foreign Key
Must be unique	May repeat many times

Q16. How can we identify a foreign key in a table?

PYQ model questions

Look for an attribute whose name or meaning matches a primary key of another relation and whose values all come from that key's value set. Confirm it by checking the declared constraint (REFERENCES clause), by the fact that it repeats across rows, and by the fact that it may be NULL. For example, DeptID in EMPLOYEE repeats and every value appears in DEPARTMENT.DeptID, so it is a foreign key.

Q17. What are relational model constraints? Name the main types.

PYQ R25 Q1(e)

Relational model constraints are the rules that every valid instance of the database must satisfy. The main types are **domain constraints**, **key constraints**, **entity integrity constraints** and **referential integrity constraints**; semantic or business rules are enforced separately by triggers or assertions.

Q18. Explain the domain constraint.

A **domain constraint** states that the value of every attribute must belong to the domain of that attribute — the correct data type, the allowed range and the permitted format. Example: Age must be an integer between 0 and 120, and Month must be one of January to December. A value outside the domain is rejected.

Q19. Explain the entity integrity constraint.

The **entity integrity constraint** states that the primary key of a relation cannot be NULL, because a tuple must be identifiable. Example: SID in STUDENT can never be NULL, whereas a non-key attribute such as Phone may be NULL.

Q20. Explain the referential integrity constraint.

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

The **referential integrity constraint** states that a foreign key value in a referencing relation must either equal some existing primary key value in the referenced relation or be wholly NULL. Example: every DeptID in EMPLOYEE must exist in DEPARTMENT; a value D99 that is not in DEPARTMENT is rejected. The constraint is enforced on insert, update and delete.

Q21. Explain the key constraint.

A **key constraint** states that no two tuples of a relation may have the same value for the primary key — the key value must be unique. It is what makes identification of a tuple possible, and it is enforced by the DBMS on every insertion and update.

Q22. What is a schema diagram?

PYQ model questions

A **schema diagram** is a picture of the database schema in which each relation is drawn as a box listing its attributes, the primary key is underlined, and arrows are drawn from each foreign key to the primary key it references. It shows structure only — no data values.

Q23. What is a descriptive attribute? Give an example.

PYQ R25 model, DBE_UNIT_2_2_1

A **descriptive attribute** is an attribute that belongs to a relationship rather than to any single entity. Example: Grade in the relationship ENROLLS between STUDENT and COURSE — the grade belongs to the particular student-course pair, not to the student alone or the course alone. Hours in WORKS_ON and Start_date in MANAGES are similar.

Q24. What is the significance of NULL attributes in a database?

PYQ R25 Q1(f), model questions

NULL marks a value that is unknown, not applicable or not yet supplied — for example the Date_of_return of a book that has not been returned. It matters because NULL never equals anything, not even another NULL, so comparisons need IS NULL; aggregates ignore NULLs; and arithmetic with NULL yields NULL. Allowing NULL makes optional facts storable without inventing dummy values, but it complicates queries and constraint checking.

Q25. What is RAID? Why is it used?

RAID (Redundant Array of Independent Disks) is a storage technique that uses several disks together to improve reliability, performance or both. Mirroring (RAID 1) duplicates data on two disks so that one can fail; striping (RAID 0) spreads data across disks for speed; parity schemes (RAID 4 and 5) store check information so that data can be reconstructed after a single disk failure.

Q26. What is an index? Why are indices used?

An **index** is an auxiliary data structure that stores search-key values together with pointers to the records containing them, so that records can be found without scanning the whole file. Indices are used to speed up selection and join operations and to enforce uniqueness, at the cost of extra storage and slower updates.

Q27. What is a B+ tree?

A **B+ tree** is a balanced multi-level index in which all search keys appear in the leaf nodes, the leaves are linked in order to allow efficient range scans, and the internal nodes contain only keys for routing. All leaves are at the same depth, so every search takes the same number of disk accesses, which makes it the standard index structure in commercial DBMSs.

Q28. Differentiate primary, secondary and clustering index.

Primary Index	Clustering Index	Secondary Index
Built on the ordering key field of an ordered file	Built on a non-key ordering field of an ordered file	Built on any non-ordering field
One entry per block	One entry per distinct value	One entry per record, or per value with a pointer list
Data file must be sorted on the key	Data file must be sorted on that field	Data file need not be sorted
At most one per file	At most one per file	Many per file

Q29. Compare sequential, indexed and hashed file organisation.

Organisation	How records are placed	Best for
Sequential (heap or sorted)	Records stored one after another in order of arrival or of a key	Full-file scans and batch reports
Indexed	Records stored with an index on a search key	Point queries and range queries
Hashed	A hash function on a key computes the block address	Exact-match queries, very fast

Q30. List the steps of mapping an ER model to the relational model.

PYQ R25 model questions

1. Map each strong entity set to a table, using its key as the primary key. 2. Map each weak entity set to a table that includes the owner's key. 3. Map each 1:1 relationship by merging or by a foreign key. 4. Map each 1:N relationship by putting the key of the one side into the many side. 5. Map each M:N relationship to a separate table. 6. Map multivalued attributes to separate tables. 7. Map each subclass according to the chosen specialization strategy.

Q31. What is meant by detailed storage architecture?

Syllabus Unit II

Storage architecture is the internal (physical) level of the DBMS: the storage hierarchy from cache and main memory down to magnetic disk and tape, the organisation of records into blocks and files, the access methods — sequential, indexed and hashed — the index structures such as B and B+ trees, and the use of RAID for reliability and speed. It determines how quickly the logical operations can actually be executed.

Q32. Define relation cardinality and give an example.

The **cardinality** of a relation is its number of tuples. If STUDENT currently holds 45 rows, its cardinality is 45. Cardinality is a property of the instance and changes over time, unlike the degree, which is fixed by the schema.

Q33. Why is the relational model called a record-based model?

Because data is represented as fixed-format **records** of a known number of fields — that is, as tables of tuples — rather than as objects or trees. The three record-based models are the relational, network and hierarchical models.

Q34. Can a primary key consist of more than one attribute? Explain.

Model question — Unit 2

Yes. When no single attribute is unique, the primary key may be **composite**, made of two or more attributes taken together. Example: in ORDER_ITEM(OrderID, ProductID, Quantity), neither OrderID nor ProductID alone is unique, but the pair (OrderID, ProductID) is, so that pair is the composite primary key. Every M:N relationship and every weak entity produces a composite key in this way.

Q35. Why can Email be an alternate key even though it uniquely identifies a customer?

Model question — Unit 2

Because uniqueness alone makes an attribute a **candidate key**, and a relation may have several. When one of them is chosen as the primary key, the rest become alternate keys. Email still enforces uniqueness and still gives a fast lookup path, so it is retained with a UNIQUE constraint rather than discarded.

Q36. Why must a candidate key be minimal while a super key need not be?

Model question — Unit 2

A candidate key is the smallest set that can identify a tuple. If it contained an extra attribute, that attribute would be redundant — it would take storage and would have to be maintained without adding any identifying power. A super key is any identifying set, so it may contain such redundant attributes; every superset of a candidate key is a super key but not a candidate key.

Q37. Can a composite primary key contain two foreign keys? Explain with an example.

Model question — Unit 2

Yes, and this is the normal case for an associative relation. In ENROLLMENT(StudentID, CourseID, Grade), the composite primary key (StudentID, CourseID) is made of two foreign keys — StudentID referencing STUDENT and CourseID referencing COURSE. The same happens in ORDER_ITEM(OrderID, ProductID) and in any table created from an M:N relationship.

Q38. What happens if a foreign key value does not exist in the referenced relation? Which constraint is violated?

Model question — Unit 2

The **referential integrity constraint** is violated and the DBMS rejects the operation. Example: inserting an EMPLOYEE with DeptID = D9 when DEPARTMENT has no D9 fails. The value is permitted only if it matches an existing primary key value or is NULL.

Q39. What is ordered indexing?

Model question — Unit 2

An **ordered index** keeps its index entries sorted on the search key, so that a binary search can be used on the index itself. A primary index, a clustering index and a secondary index on an ordered field are all ordered indexes. They support both exact-match and range queries, unlike a hash index, which supports only exact match.

Q40. Explain the basic organisation of magnetic disk storage.

Model question — Unit 2

A magnetic disk consists of several **platters** with **tracks** on each surface, and each track is divided into **sectors**. A read/write head moves to the correct track (seek time), waits for the required sector to rotate under it (rotational delay), and then transfers the data. The three delays together — seek, rotational and transfer — decide the cost of one block access, which is why the DBMS tries to minimise the number of blocks read.

Q41. Explain record organisation in a database.

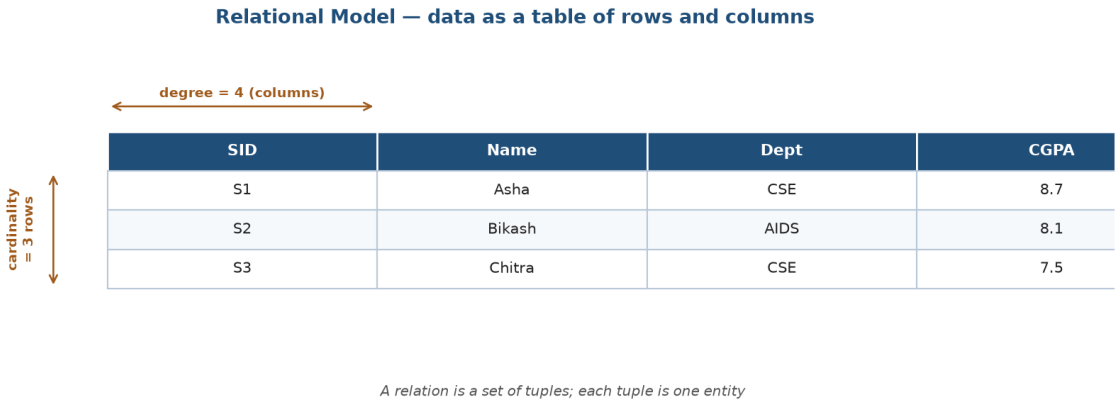
Model question — Unit 2

A record is a collection of fields, and records are grouped into blocks or pages, which are the unit of transfer between disk and memory. Records may be **fixed length**, where each record occupies the same number of bytes, or **variable length**, where a length indicator or a separator is used. A **spanned** organisation allows one record to cross a block boundary; an **unspanned** organisation keeps each record inside one block.

Section B — 5-Mark Questions

Q1. Explain the terms relation, tuple, attribute, domain, degree and cardinality.

5 marks

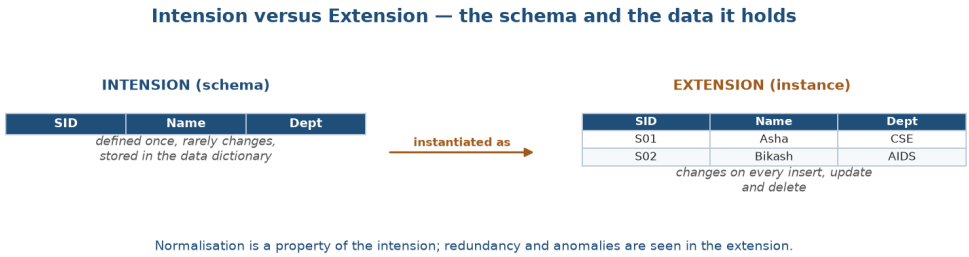


Term	Meaning	Example
Relation	A table of values; formally a subset of the Cartesian product of domains	STUDENT
Tuple	One row — one entity	{S01, Asha, CSE, 8.7}
Attribute	One named column — one property	Name, Dept
Domain	The set of atomic values an attribute may take	Age: integers 0 to 120
Degree (arity)	The number of attributes, that is the columns	4
Cardinality	The number of tuples, that is the rows	3

The degree is fixed by the schema and never changes; the cardinality changes on every insert and delete. Because a relation is a **set**, duplicate tuples are not permitted and the order of tuples is not significant.

Q2. Explain schema, instance, intension and extension with examples.

5 marks

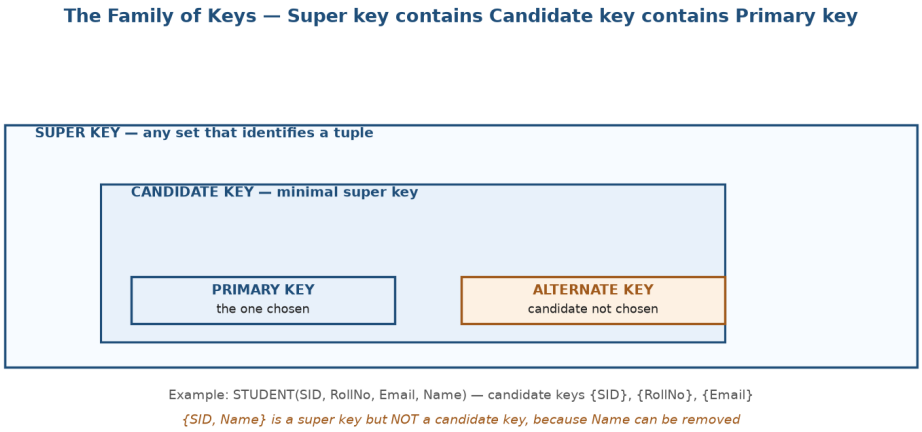


Intension (schema)	Extension (instance)
The permanent, structural part of a relation	The changing part — the tuples currently stored
Name, attributes, domains and constraints	The actual data values
Defined at design time using DDL	Populated and modified using DML
Stored in the data dictionary	Stored in the data files
Changes rarely	Changes constantly

Example: the schema STUDENT(SID, Name, Dept) is the intension; the rows {S01, Asha, CSE} and {S02, Bikash, AIDS} are the extension. Normalisation is a property of the intension, while redundancy and anomalies are observed in the extension.

Q3. Explain super key, candidate key and primary key with an example.

5 marks



Super key

Any set of attributes that uniquely identifies a tuple. In STUDENT(SID, RollNo, Email, Name), {SID}, {SID, Name} and {RollNo, Email} are all super keys.

Candidate key

A **minimal** super key, from which no attribute can be removed. Here the candidate keys are {SID}, {RollNo} and {Email}.

Primary key

The candidate key chosen by the designer. Suppose SID is chosen; it must be unique and can never be NULL.

The relationship

Every primary key is a candidate key, every candidate key is a super key, but not the reverse. If a relation has n attributes and a candidate key of size k, that key alone generates 2^{n-k} super keys.

Q4. Explain alternate, composite and foreign keys with examples.

5 marks

Key	Meaning	Example
Alternate key	A candidate key not chosen as the primary key	RollNo, Email in STUDENT
Composite key	Two or more attributes together form the key	(OrderID, ProductID) in ORDER_ITEM
Foreign key	References the primary key of another relation	DeptID in STUDENT referencing DEPARTMENT
Partial key	Discriminator of a weak entity set	Dependent_name in DEPENDENT

The alternate key still enforces uniqueness and gives a second access path. The composite key is used when no single attribute is unique. The foreign key is how relationships are represented in the relational model, and it is what the referential integrity constraint is checked against.

Q5. Explain domain and key constraints with examples.

5 marks

Domain constraint

Every value must belong to the attribute's domain — the correct data type, the allowed range and the permitted format. Example: Age must be an integer between 0 and 120; Month must be one of January to December. It is enforced by the declared data type together with CHECK clauses.

Key constraint

No two tuples may have the same primary key value. Example: two rows of STUDENT cannot both have SID = S01. It is declared with PRIMARY KEY and checked on every insertion and update.

Why both are needed

The domain constraint protects a single value; the key constraint protects the identity of a whole tuple. Without the first, garbage values enter; without the second, the same entity can be stored twice and can no longer be identified.

Q6. Explain entity integrity and referential integrity with examples.

5 marks

Entity integrity

The primary key of a relation can never be NULL, because a tuple whose key is unknown cannot be identified. Example: SID in STUDENT can never be NULL, while a non-key attribute such as Phone may be.

Referential integrity

A foreign key value must either equal some existing primary key value in the referenced relation, or be wholly NULL. Example: every DeptID in EMPLOYEE must exist in DEPARTMENT; a value D9 that is not there is rejected.

Where they are checked

Both are checked on insert and update. Referential integrity is additionally checked when a parent row is deleted or its key is changed, which is what the referential actions govern.

Q7. Explain the referential actions available when a parent row changes.

5 marks

Action	Effect
RESTRICT	The operation is rejected if child rows exist
CASCADE	The same delete or update is applied to the child rows
SET NULL	The child foreign key is set to NULL
SET DEFAULT	The child foreign key is set to a default value
NO ACTION	The check is deferred and fails only if the constraint is still violated at the end

RESTRICT is the safe default because it never destroys data silently. CASCADE suits genuinely dependent data, such as order lines that cannot exist without their order. SET NULL suits an optional relationship, where the child may simply lose its parent reference.

Q8. Explain the mapping of a strong entity set with composite and multivalued attributes.

5 marks

ER construct	Tables	How
Strong entity set	1	Key becomes the primary key, simple attributes become columns
Composite attribute	still 1	Store only the simple components; the composite itself disappears

ER construct	Tables	How
Multivalued attribute	2	One table for the entity, one holding the key and the values

Example

Student with Name (First_name, Last_name) and Address (House_no, Street, City) becomes Student(Roll_no, First_name, Last_name, House_no, Street, City). Adding a multivalued Phone_no gives a second table Student_Phone(Roll_no, Mobile_no) whose primary key is the pair.

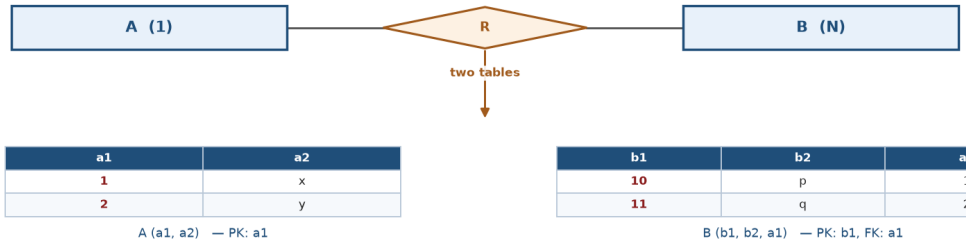
Why

The relational model allows only atomic values, so a composite attribute has no representation of its own, and a set of values cannot be stored in one column without violating 1NF.

Q9. Explain the mapping of 1:1, 1:N and N:1 relationships.

5 marks

Mapping a 1:N relationship — the key moves to the MANY side



a1	a2
1	x
2	y

A (a1, a2) — PK: a1

b1	b2	a1
10	p	1
11	q	2

B (b1, b2, a1) — PK: b1, FK: a1

The relationship is merged into B, the many side; a1 becomes a foreign key there.

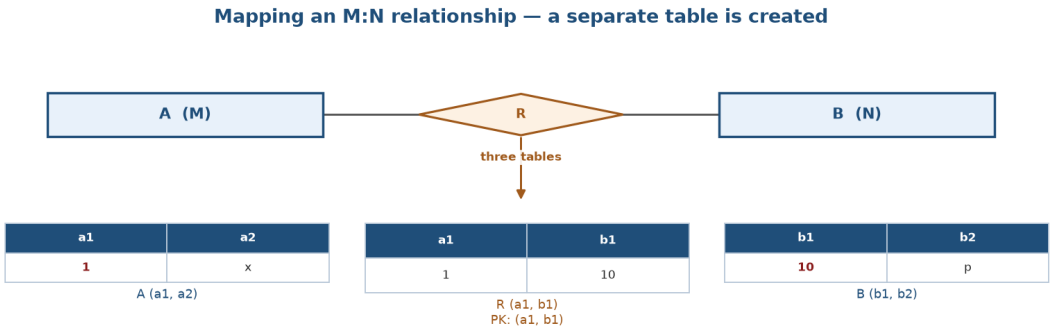
Cardinality	Tables	Rule
1:N or N:1	2	Merge the relationship into the many side; the key of the one side becomes a foreign key there
1:1, both partial	2	Place the foreign key on either side
1:1, one side total	2	Place the foreign key on the total side
1:1, both total	1	Merge everything into a single table and take either key

Example

DEPARTMENT 1 : N EMPLOYEE gives EMPLOYEE(SSN, Name, ..., Dnumber) with Dnumber as a foreign key. Because every employee must belong to a department, Dnumber is also NOT NULL — total participation on the many side turns the merged foreign key into a mandatory column.

Q10. Explain the mapping of M:N relationships and weak entity sets.

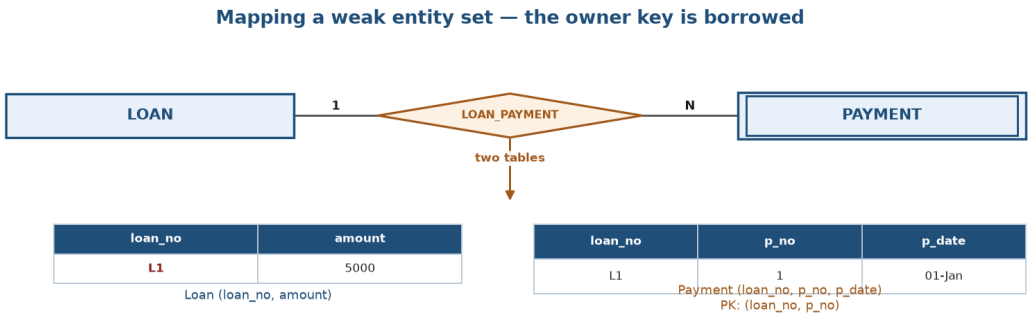
5 marks



The new table's primary key is the combination of both keys; descriptive attributes go here too.

M:N relationship

Three tables: the two entity tables plus one for the relationship, whose primary key is the combination of both keys and which also holds the descriptive attributes. Example: ENROLLS(Student_id, Course_no, Grade) between STUDENT and COURSE.



The weak entity's primary key = owner key + its own partial key.

Weak entity set

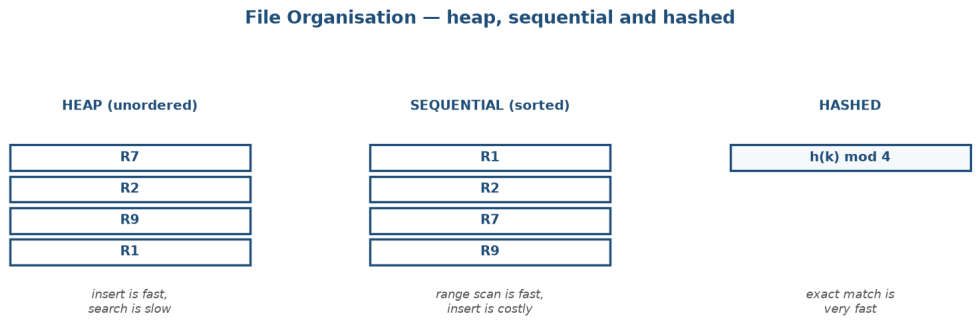
Two tables: the owner table and the weak table, whose primary key is the owner's key plus the partial key. Example: LOAN(Loan_no, Amount) and PAYMENT(Loan_no, P_no, P_date), keyed by (Loan_no, P_no).

Why neither can be merged

An M:N relationship has no single place in either table for the foreign key, and a weak entity has no key of its own to identify its rows.

Q11. Explain file organisation — sequential, indexed and hashed.

5 marks

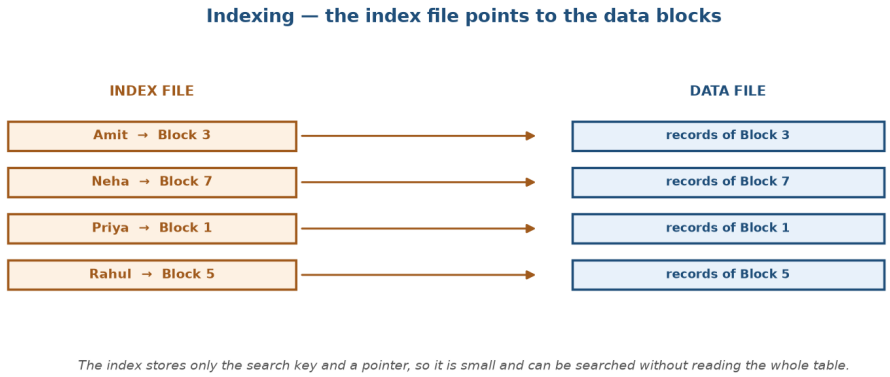


Organisation	Placement	Advantage	Disadvantage
Heap (unordered)	Anywhere with free space	Fast insert	Search needs a full scan
Sequential (sorted)	In order of a key	Fast range scan and ordered output	Costly insert and delete
Hashed	Address computed by a hash function	Very fast exact match	Poor for range queries
Clustered	Related records of several files kept together	Fast join	Reorganisation needed as data grows

A heap suits a table that is mostly inserted into, a sorted file suits reporting on a key, and hashing suits lookup by an exact key such as an account number. Real DBMSs usually keep the data in a heap and add indexes for access.

Q12. Explain primary, secondary and clustering indexes.

5 marks



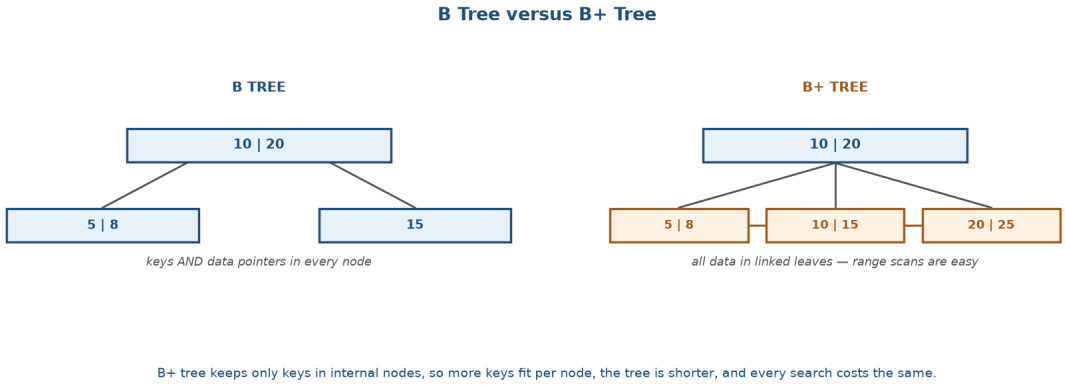
Type	Built on	Entries	Per file
Primary index	The ordering key field of an ordered file	One per block	One
Clustering index	A non-key ordering field of an ordered file	One per distinct value	One
Secondary index	Any non-ordering field	One per record, or one per value with a pointer list	Many

A primary index is the smallest and fastest because it is sparse and the file is already sorted on that key. A clustering index handles duplicates of a non-key ordering field. A secondary index is needed for every other column that must be searched, and it is larger because the file is not sorted on that

field.

Q13. Compare B tree and B+ tree.

5 marks



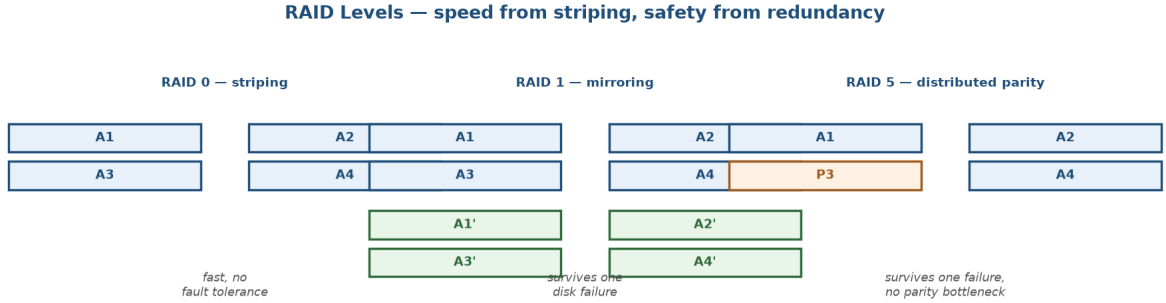
B tree	B+ tree
Keys and data pointers in every node	Keys only in internal nodes; all data pointers in the leaves
No linked leaves	Leaves linked in order, so range scans are easy
A search may end at an internal node	Every search reaches a leaf
Lower fan-out, taller tree	Higher fan-out, shorter tree

Because internal nodes of a B+ tree hold only keys, more keys fit in one block, so the tree has fewer levels and fewer disk accesses per search. Every search costs the same number of accesses, and the linked leaves make range queries efficient — which is why the B+ tree is the standard index structure in commercial DBMSs.

Q14. Explain the storage hierarchy and the RAID levels used for databases.

5 marks

Level	Characteristics
Cache and main memory	Fastest, volatile, small
Flash memory	Fast, non-volatile, moderate size
Magnetic disk	Slower, non-volatile, large
Optical and tape	Archival, very slow, very large



RAID level	Technique	Benefit
RAID 0	Block striping, no redundancy	Maximum speed, no fault tolerance
RAID 1	Mirroring	Survives one disk failure, costs half the space
RAID 5	Block striping with distributed parity	Survives one failure, no parity bottleneck
RAID 6	Two independent parities	Survives two failures

A database server uses RAID 1 or RAID 5 because durability requires that a disk failure not destroy committed data, while performance requires that reads be spread across several disks.

Section C — 7-Mark Questions

Q1. Discuss all the types of keys in a relational database with suitable examples and explain their role in data integrity.

PYQ R25 Q3(c), B26 Q3(a), R24 Q3(a) — a 7-mark question in all three papers

Super key

Any set of attributes that uniquely identifies a tuple. In STUDENT(SID, RollNo, Name, Dept, Email) where SID, RollNo and Email are each unique, {SID}, {SID, Name}, {SID, RollNo} and {SID, RollNo, Name, Email} are all super keys.

Candidate key

A minimal super key, from which no attribute can be dropped. Here the candidate keys are {SID}, {RollNo} and {Email}.

Primary key

The candidate key chosen to identify tuples. Suppose SID is chosen. It must be unique and never NULL.

Alternate key

The candidate keys not chosen as primary — here RollNo and Email.

Foreign key

An attribute in one relation referencing the primary key of another. DeptID in STUDENT referencing DEPARTMENT(DeptID).

Composite key

A key of two or more attributes, used when no single attribute is unique. In ORDER_ITEM(OrderID, ProductID, Qty), the pair (OrderID, ProductID) is the composite primary key.

Partial key

The discriminator of a weak entity set, such as Dependent_name, which identifies an entity only within its owner.

Role in data integrity

Key	Integrity role
Super key	Provides the pool of all identifying attribute sets
Candidate key	Guarantees at least one minimal identifier exists
Primary key	Enforces entity integrity — unique and not NULL
Alternate key	Gives a second unique access path, useful for lookups
Foreign key	Enforces referential integrity — links relations without pointers
Composite key	Allows identification where only a combination is unique

Worked drill — counting super keys

For a relation with three single-attribute candidate keys A, B, C and two other attributes X, Y, every superset of a candidate key is a super key. From {A} alone there are $2^2 = 4$ supersets ({A}, {A,X}, {A,Y}, {A,X,Y}); with three candidate keys, allowing for the overlaps, the count is $3 \times 4 - 3 = 9$ distinct super keys. This is the standard way to answer "list at least eight super keys".

Q2. Explain the rules for converting an ER diagram into relational tables, with examples.

PYQ R25 Q3(b), B26 Q3(c), R24 Q3(b), Q3(c)

Rule 1 — Strong entity set

Each strong entity set becomes one table. Its key attribute becomes the primary key and its simple attributes become columns.

Rule 2 — Composite attributes

A strong entity set with composite attributes still needs only **one** table: the simple components are stored and the composite attribute itself is not. Example: Student(Roll_no, First_name, Last_name, House_no, Street, City).

Rule 3 — Multivalued attributes

A strong entity set with multivalued attributes needs **two** tables: one with the simple attributes and the primary key, and one holding the primary key plus the multivalued attribute. Example: Student(Roll_no, City) and Student_Phone(Roll_no, Mobile_no).

Rule 4 — Relationship set as its own table

A relationship set can become one table whose attributes are the primary keys of the participating entity sets plus its own descriptive attributes; the combination of the participating keys is the primary key. Example: Works_in(Emp_no, Dept_id, since).

Rule 5 — Binary relationships by cardinality ratio

Cardinality	Tables required	How
M:N	3	A(a1,a2), R(a1,b1), B(b1,b2)
1:N	2	A(a1,a2) and BR(a1,b1,b2) — merge R into the N-side entity B
N:1	2	AR(a1,a2,b1) and B(b1,b2) — merge R into the N-side entity A
1:1	2	Either AR(a1,a2,b1)+B or A+BR(a1,b1,b2)

Rule 6 — The three cases of a 1:1 relationship

If both sides are partial, the foreign key may be placed on either side. If one side is total, the foreign key is always placed on the **total** side. If both sides are total, the two entity sets and the relationship are merged into a **single** table, and either key may be chosen as the primary key.

Rule 7 — Total participation

When a relationship is merged into an entity set because of a 1:N or N:1 cardinality, and the participation on that side is total, the merged foreign key acquires a **NOT NULL** constraint, because it can never be empty.

Rule 8 — Weak entity set

A weak entity set always appears with an identifying relationship and total participation. It needs **two** tables: the owner table A(a1,a2) and the weak table BR(a1,b1,b2), where the primary key is the owner's key plus the partial key. Example: Loan(Loan_no, Amount) and Payment(Loan_no, P_no, P_date, P_amount).

Rule 9 — Specialization

There are three strategies: a single table for the superclass with a type attribute and all subclass attributes; one table per subclass containing only the subclass attributes plus the superclass key; or one table per subclass containing all inherited attributes. The choice depends on whether the specialization is disjoint or overlapping and total or partial.

Q3. Explain all the relational model constraints with examples.

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

Domain constraint

Each value must come from the attribute's domain. Example: Salary must be numeric and non-negative; Gender must be one of a permitted list. Enforced at insertion by type and CHECK clauses.

Key constraint

No two tuples may share the same primary key value. Example: two rows in STUDENT cannot both have SID = S01.

Entity integrity constraint

The primary key may not be NULL. Example: SID cannot be NULL, because then the tuple could not be identified.

Referential integrity constraint

A foreign key value must exist in the referenced relation or be NULL. Example: inserting an EMPLOYEE with DeptID = D9 when DEPARTMENT has no D9 is rejected.

Semantic integrity constraints

Business rules that the schema cannot express directly, enforced with assertions or triggers. Example: an employee's salary may not exceed the manager's salary; a student may register for at most six courses.

Effect of each operation

Operation	Constraint checked
INSERT	Domain, key, entity integrity, referential integrity
UPDATE of key	Key and referential integrity (in the child rows too)
UPDATE of foreign key	Referential integrity
DELETE of a parent	Referential integrity — restrict, cascade or set NULL

Referential actions on delete

RESTRICT rejects the delete if children exist; **CASCADE** deletes the children as well; **SET NULL** makes the children's foreign key NULL; **SET DEFAULT** assigns a default value.

Q4. Find the candidate keys and the primary key for the given relations, and list super keys.

PYQ model questions, DBMS_Keys_10_More_Problems, super_key_candidate_key

Method

Use the closure method. An attribute that never appears on the right-hand side of any functional dependency must be part of every candidate key. Start with such attributes, compute the closure, and add attributes only until the closure equals the whole relation; then check minimality.

Drill 1 — STUDENT(StudentID, RollNo, Name, Email, Department)

StudentID, RollNo and Email are each individually unique, so each is a candidate key. If StudentID is chosen as the primary key, RollNo and Email become alternate keys. Super keys include

{StudentID}, {StudentID, Name}, {RollNo}, {RollNo, Department}, {Email, Name} and every other superset of a candidate key.

Drill 2 — ORDER_ITEM(OrderID, ProductID, Quantity, UnitPrice)

A product occurs at most once in an order, so OrderID alone repeats and ProductID alone repeats, but the pair is unique. Candidate key is {OrderID, ProductID}. It is composite because neither attribute is unique by itself.

Drill 3 — FLIGHT(FlightNo, FlightDate, Source, Destination, AircraftNo)

The same flight number operates on many dates, so FlightNo alone is not unique. The candidate key is {FlightNo, FlightDate}.

Drill 4 — ROOM_BOOKING(RoomNo, BookingDate, TimeSlot, CustomerID)

A room can be booked many times in a day but not in the same slot. The candidate key is {RoomNo, BookingDate, TimeSlot}.

Drill 5 — BANK_ACCOUNT(AccountNo, CustomerID, BranchCode, AccountType, Balance)

AccountNo uniquely identifies each account, so {AccountNo} is the candidate key and the primary key. CustomerID and BranchCode are foreign keys to CUSTOMER and BRANCH.

Drill 6 — ENROLLMENT(StudentID, CourseID, Semester, Grade)

A student may register for many courses and a course may have many students, so the composite primary key is {StudentID, CourseID, Semester}.

Why minimality matters

A super key may contain redundant attributes, which would allow the same fact to be recorded twice; a candidate key is minimal, so it identifies a tuple without storing anything unnecessary. Every superset of a candidate key is a super key, which is why the count of super keys grows as 2 to the power of the remaining attributes.

Q5. Explain the storage architecture of a DBMS: storage hierarchy, file organisation, indexing and RAID.

Syllabus Unit II — detailed storage architecture

Storage hierarchy

Data moves between levels: **cache and main memory** (fast, volatile, small), **flash memory** (fast, non-volatile), **magnetic disk** (slower, non-volatile, large) and **optical or tape** (archival). The DBMS tries to keep frequently used blocks in memory and to minimise disk accesses, because disk I/O dominates cost.

Records, blocks and files

A **record** is a collection of fields; records are grouped into **blocks** (pages), which are the unit of transfer between disk and memory. A **file** is a sequence of records, and its **file organisation** decides where each record is placed.

File organisations

Organisation	Placement	Advantage	Disadvantage
Heap (unordered)	Anywhere with free space	Fast insert	Slow search, needs full scan
Sequential (sorted)	In order of a key	Fast range scan and ordered output	Costly insert and delete
Hashed	Address computed by hash function	Very fast exact-match lookup	Poor for range queries

Organisation	Placement	Advantage	Disadvantage
Clustered	Related records of several files stored together	Fast join	Reorganisation needed when data grows

Indexing

A **primary index** is built on the ordering key field, one entry per block. A **clustering index** is built on a non-key ordering field, one entry per distinct value. A **secondary index** is built on a non-ordering field and may be dense, with one entry per record. A **multilevel index** indexes the index itself, which is the idea behind the B tree and B+ tree.

B tree versus B+ tree

B Tree	B+ Tree
Keys and data pointers in every node	Keys only in internal nodes; all data pointers in leaves
No linked leaves	Leaves linked in order for range scans
Search may end at an internal node	Every search reaches a leaf
Less fan-out, taller tree	Higher fan-out, shorter tree

RAID

Level	Technique	Benefit
RAID 0	Block striping, no redundancy	Maximum speed, no fault tolerance
RAID 1	Mirroring	Survives one disk failure, 50 percent space cost
RAID 4	Block striping with one parity disk	Survives one failure, parity disk bottleneck
RAID 5	Block striping with distributed parity	Survives one failure, no bottleneck
RAID 6	Two independent parities	Survives two failures

Q6. Explain ER-to-relational mapping for a hospital database and for a car-insurance database.

Mapping examples from the class notes

Hospital database

Entities: PATIENT, DOCTOR, TEST (a weak entity of PATIENT through the identifying relationship TEST_LOG), and the relationships DR_PATIENT and PERFORMED_BY.

Table	Attributes	Key and constraints
patients	patient_id, name, insurance, date_admitted, date_checked_out	PK: patient_id
doctors	doctor_id, name, specialization	PK: doctor_id
test	test_id, test_name, date, time, result, patient_id	PK: (patient_id, test_id); FK: patient_id
doctor_patient	patient_id, doctor_id	PK: (patient_id, doctor_id); both FKs
performed_by	test_id, doctor_id	PK: (test_id, doctor_id); both FKs

The weak entity TEST is given the owner's key patient_id as part of its primary key, which is exactly Rule 8.

Car-insurance database

Entities: PERSON, CAR, ACCIDENT, with PERSON owning CAR (1:N) and PERSON, CAR participating in ACCIDENT through the M:N relationship PARTICIPATED, which carries Damage_amount.

Table	Attributes	Key and constraints
person	driver_id, name, address	PK: driver_id
car	license, year, model, driver_id	PK: license; FK: driver_id to person
accident	report_number, date, location	PK: report_number
participated	driver_id, license, report_number, damage_amount	PK: (driver_id, license, report_number); three FKs

The 1:N relationship between PERSON and CAR is merged into the many side, so driver_id appears inside car. The M:N relationship PARTICIPATED becomes its own table because it carries a descriptive attribute.

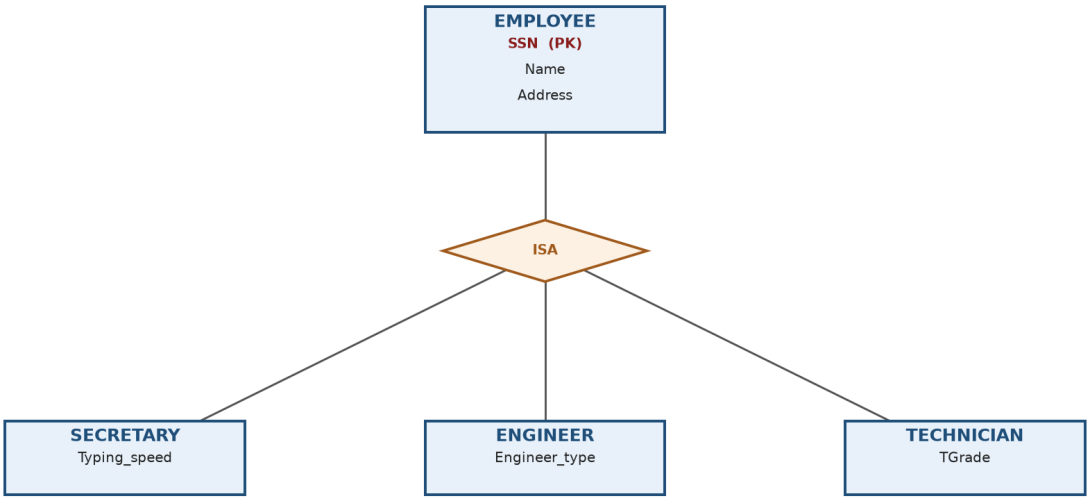
Q7. Explain specialization and how it is mapped to relational tables.

PYQ R24 Q3(b)

The concept

Specialization divides a superclass into subclasses. In EER, EMPLOYEE specialises into SECRETARY, ENGINEER and TECHNICIAN, each with its own attributes, and each subclass inherits all the superclass attributes and relationships.

Generalization / Specialization — EER Notation



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

Strategy 1 — Single table

One table EMPLOYEE(SSN, Name, Address, Typing_speed, Engineer_type, TGrade, Emp_type). The type attribute Emp_type says which subclass the row belongs to. This is good when the subclasses have few specific attributes and the specialization is overlapping, but it wastes space with many NULLs.

Strategy 2 — Table per subclass, superclass key only

EMPLOYEE(SSN, Name, Address) plus SECRETARY(SSN, Typing_speed), ENGINEER(SSN, Engineer_type) and TECHNICIAN(SSN, TGrade). Each subclass table's primary key is also a foreign key to EMPLOYEE. This works for any constraint but needs joins to rebuild a full entity.

Strategy 3 — Table per subclass with all attributes

SECRETARY(SSN, Name, Address, Typing_speed), ENGINEER(SSN, Name, Address, Engineer_type) and TECHNICIAN(SSN, Name, Address, TGrade). No join is needed, but common attributes are repeated and a disjoint-total constraint cannot be enforced without extra assertions.

Choosing

Situation	Best strategy
Total and disjoint specialization, few subclass attributes	Single table
Many subclass-specific attributes, any constraint	Table per subclass with superclass key
Queries mostly read one subclass	Table per subclass with all attributes
Overlapping specialization	Single table or table per subclass with key

Q8. Compare relation intension and extension, schema and instance, with examples.

PYQ model questions

Intension versus extension

The **intension** of a relation is its schema — name, attributes, domains and constraints. The **extension** is the set of tuples stored at a given time. Intension is defined once with DDL and rarely changes; extension changes with every DML operation.

Aspect	Intension (Schema)	Extension (Instance)
What it is	Structural description	Actual data rows
When fixed	At design time	Moment to moment
Defined by	DDL	DML
Changes	Rarely	Frequently
Example	STUDENT(SID, Name, Dept)	{S01, Asha, CSE}, {S02, Bikash, AIDS}
Stored in	Data dictionary	Data files

Why the distinction matters

The DBMS checks every new tuple against the intension — the domain, key, entity and referential constraints — before it becomes part of the extension. Normalisation is a property of the intension, while redundancy and anomalies are observed in the extension. Understanding the difference also explains data independence: the intension can be changed with little effect on programs precisely because the extension is accessed only through it.

Q9. Explain descriptive attributes and NULL attributes with examples, and their effect on design.

PYQ R25 Q1(f), model questions

Descriptive attributes

A descriptive attribute belongs to a **relationship instance**. In the COMPANY schema, Hours describes the participation of one employee in one project, and Start_date describes when one employee became manager of one department. Such attributes cannot be moved into either entity table without losing information — if Hours were stored in EMPLOYEE, we could not record different hours for different projects.

Consequence for mapping

A relationship with descriptive attributes must become its own table: WORKS_ON(Essn, Pno, Hours). This is the same rule that forces M:N relationships into separate tables.

NULL attributes

NULL means unknown, not applicable or withheld. Types of NULL are: **unknown** (the value exists but we do not know it), **not applicable** (the attribute does not apply to this entity, such as Spouse_name for an unmarried employee) and **not yet available** (it will be supplied later).

Design consequences

Issue	Effect of NULL
Keys	Primary key cannot be NULL (entity integrity)
Arithmetic	Any expression containing NULL evaluates to NULL
Comparison	NULL = NULL is unknown; must use IS NULL
Aggregates	COUNT ignores NULLs; AVG divides by the count of non-NULL values
Joins	NULL never matches, so outer joins are needed to keep unmatched rows
Storage	A NULL bitmap saves space compared with dummy values

Design advice

Minimise nullable columns by splitting optional facts into their own relation, use a default only when a genuine default exists, and never use NULL to mean zero or empty string, because the two cannot then be told apart.

Q10. Consider EMPLOYEE(EmpID, AadhaarNo, Email, Name, Salary) and find all candidate keys, the primary key, alternate keys and six super keys.

Model question — Unit 2

The relation

EmpID, AadhaarNo and Email are each individually unique, so each is a candidate key.

Item	Answer
Candidate keys	{EmpID}, {AadhaarNo}, {Email}
Primary key	{EmpID}, by choice
Alternate keys	{AadhaarNo}, {Email}
Super keys	{EmpID}, {EmpID, Name}, {AadhaarNo}, {AadhaarNo, Salary}, {Email}, {Email, Name, Salary}

Why the extra super keys exist

Every superset of a candidate key is also a super key, because adding attributes cannot destroy uniqueness. With five attributes in all and a candidate key of size one, each candidate key generates $2^4 = 16$ super keys, so the relation has $3 \times 16 - 3 = 45$ distinct super keys.

Q11. Consider PRODUCT(ProductID, SKU, Barcode, Name, CategoryID) and identify all the keys.

Model question — Unit 2

The relation

ProductID, SKU and Barcode are each unique; Name and CategoryID repeat.

Item	Answer
Super keys	{ProductID}, {SKU}, {Barcode}, {ProductID, Name}, {SKU, CategoryID}, {Barcode, ProductID}
Candidate keys	{ProductID}, {SKU}, {Barcode}
Primary key	{ProductID}
Alternate keys	{SKU}, {Barcode}

Item	Answer
Foreign key	CategoryID references CATEGORY(CategoryID)

Why CategoryID is not a key here

Many products share one category, so CategoryID repeats and cannot identify a product. It is a foreign key, not a key of this relation.

Q12. Consider CUSTOMER(CustomerID, AadhaarNo, Email, Name, City) and explain why {CustomerID, Name} is a super key but not a candidate key.

Model question — Unit 2

The keys

Candidate keys: {CustomerID}, {AadhaarNo}, {Email}. If CustomerID is chosen as the primary key, the other two are alternate keys.

Super keys

{CustomerID}, {CustomerID, Name}, {CustomerID, City}, {AadhaarNo}, {AadhaarNo, Name}, {Email, City}, and every other superset of a candidate key.

The reason

{CustomerID, Name} identifies every tuple, because CustomerID alone already does — so it is a super key. But it is not **minimal**: Name can be removed and the remaining attribute still identifies the tuple. Since a candidate key must be minimal, {CustomerID, Name} is a super key only.

Q13. Consider EMPLOYEE works_for DEPARTMENT and analyse the relationship after mapping.

Model question — Unit 2

The scenario

Each employee works for exactly one department, and one department can have many employees.

Aspect	Answer
Cardinality ratio	1 : N from DEPARTMENT to EMPLOYEE
Participation of EMPLOYEE	Total — every employee must belong to a department
Participation of DEPARTMENT	Partial — a newly created department may have no employee yet
Primary key of EMPLOYEE	EmpID
Primary key of DEPARTMENT	DeptID
Foreign key after mapping	DeptID inside EMPLOYEE

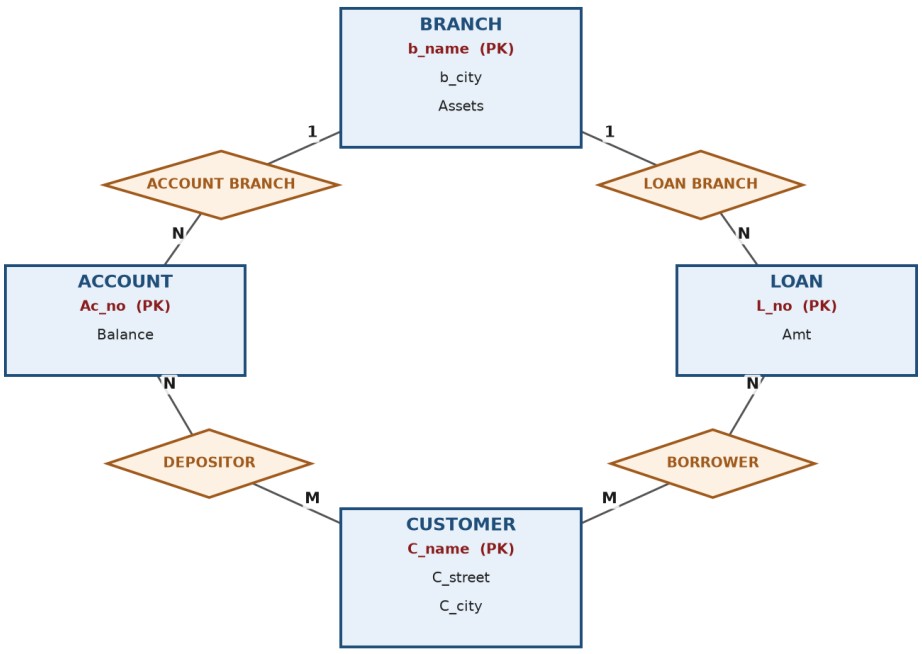
After mapping

EMPLOYEE(EmpID, Name, Salary, DeptID) with primary key EmpID and foreign key DeptID referencing DEPARTMENT. Because the employee side participates totally, DeptID also carries a NOT NULL constraint.

Q14. Explain the mapping of a bank ER diagram to relational tables.

PYQ B26 Q3(c), R24 Q3(c)

BANKING Database — ER Diagram (repeated in PYQ 2025 and 2026)



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

The diagram

BRANCH with b_name, b_city, Assets; ACCOUNT with Ac_no, Balance; LOAN with L_no, Amt; CUSTOMER with C_name, C_street, C_city. ACCOUNT_BRANCH and LOAN_BRANCH are 1:N relationships from BRANCH, and DEPOSITOR and BORROWER are M:N relationships with CUSTOMER.

Table	Attributes	Key and constraints
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

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 with a composite key.