Database Management Systems

Relational model and keys

The notes for the lesson “Relational model and keys”: chapters 2.1 to 2.2, with their figures, worked examples, exam traps, quizzes and exam question bank, in one file.

2 chapters | Chapters 2.1 to 2.2

About these notes

These are the notes for the University free n easy video lesson **Relational model and keys** on Database Management Systems: chapters 2.1 to 2.2, with every figure, worked example, exam trap and quiz, and the long-answer exam questions on these chapters.

2.1 Relational model: relations, tuples, schema and instance

The relational model, proposed by E. F. Codd of IBM in 1970, stores all data as relations: tables with named columns in which every cell holds one value. A database is a set of such tables, and every question is answered by combining tables by their values, never by following stored pointers. This one idea separates what the data means from how it sits on disk, which is why nearly every business system, from IRCTC ticket bookings to a bank's UPI ledger, runs on a relational DBMS.

You need: An earlier lesson, Data models: hierarchical, network, relational, object-oriented; and an earlier lesson, Mapping ER diagrams to relational tables, which produces the relations studied here.

The vocabulary of a relation

Every term below has a plain table word next to it. Use the formal word in an exam answer and the table word to remember it.

FORMAL TERM	TABLE WORD	MEANING
Relation	Table	A set of tuples that all have the same attributes
Tuple	Row	One fact: one value for each attribute

FORMAL TERM	TABLE WORD	MEANING
Attribute	Column name	A named property, such as `cgpa`
Domain	Column type	The set of values an attribute may take, such as decimals from 0.0 to 10.0
Degree (arity)	Number of columns	How many attributes the relation has
Cardinality	Number of rows	How many tuples the relation holds right now

The formal definition is short. Take domains $D_1, D_2, \ldots, D_n$. Their Cartesian product $D_1 \times D_2 \times \ldots \times D_n$ is the set of every possible n-tuple $\langle v_1, v_2, \ldots, v_n \rangle$ with each v_i taken from D_i. A **relation** on these domains is any subset of that product. So a relation is simply the chosen set of tuples that are true facts today. For a tuple t and an attribute A, the notation t[A] (or t.A) means the value of A in t.

What "a relation is a set" forces

Because a relation is a set, four properties follow, and examiners ask for them by name.

1. **No duplicate tuples.** A set cannot hold the same element twice, so two identical rows can never both be in a relation.
2. **Tuple order does not matter.** Shuffling the rows gives the same relation; there is no "first row".
3. **Attribute order does not matter** when attributes are named, because a tuple is then a mapping from names to values. (In the pure mathematical definition with numbered positions, order is fixed; most textbooks use the named view.)
4. **Values are atomic.** Each cell holds one value from its domain, not a list or a nested table. This is first normal form; see a later lesson, First, second and third normal forms.

One special value, **null**, marks a value that is unknown or does not apply, such as the phone number of a student who has none. Null is not zero and not an empty string; Chapter 2.8 shows why Codd made this a rule.

Schema and instance

The **relation schema** is the design: the relation's name, its attributes and their domains, written $R(A_1, A_2, \ldots, A_n)$, for example Student(roll_no, name, dept, cgpa). The **relation instance** r(R) is the set of tuples stored at one moment. A **database schema** is the set of all relation schemas plus their constraints; a **database instance** is the set of all current relation instances.

The schema changes rarely, and only through a data definition command such as `ALTER TABLE`. The instance changes with every insert, delete and update. A programming analogy helps: in C, the declaration `int marks[60];` is like a schema, while the 60 values held in `marks` at 10:15 a.m. are like an instance. In the three-schema architecture of an earlier lesson, the schemas in this unit are logical (conceptual) schemas.

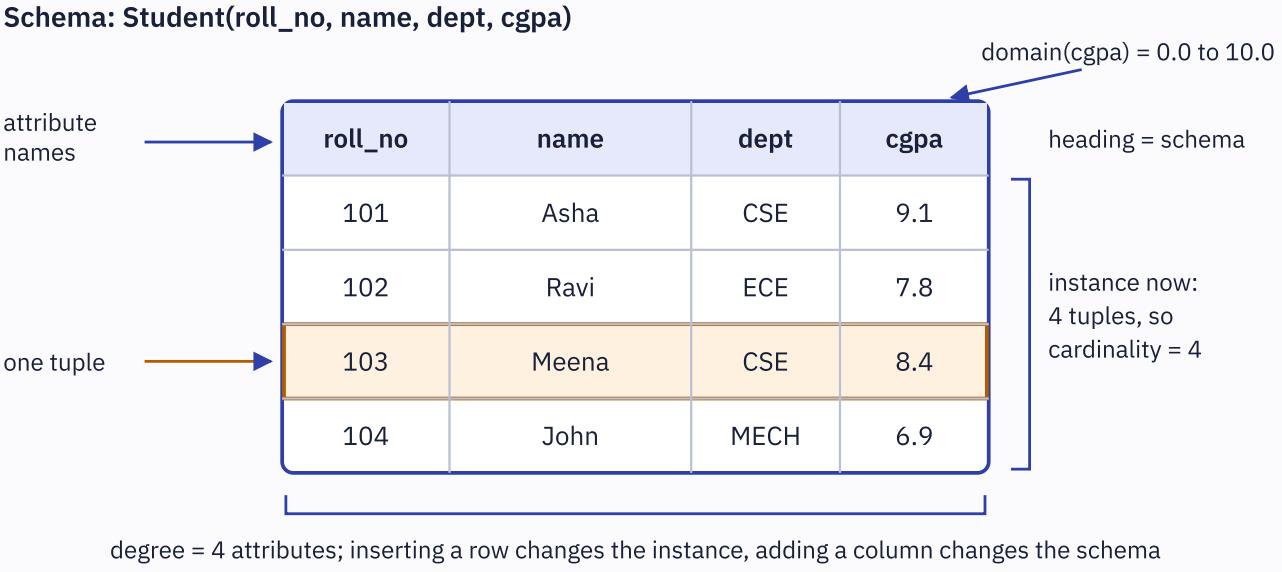


Figure 2.1.1 The Student relation. The shaded heading is the schema; the four rows below it are today's instance. Notice that degree counts columns (4) and cardinality counts rows (4 now), and that only the cardinality moves when students join or leave.

WORKED EXAMPLE

Question (6 marks). The relation Enrol(roll_no, course, grade) has domains roll_no ∈ {101, 102, 103}, course ∈ {DBMS, OS} and grade ∈ {A, B, C, D}. Its instance today is (101, DBMS, A), (101, OS, B), (102, DBMS, B), (103, OS, C).
(a) State the degree and cardinality. (b) How many different tuples are possible, and how many different instances? (c) A rule says each student takes each course at most once. What is the largest possible cardinality? (d) The grade of 103 in OS is changed to B and (102, OS, A) is inserted. Did the schema or the instance change? Give the new degree and cardinality.

1. **Degree and cardinality.** Three attributes, so degree = 3. Four tuples, so cardinality = 4.
2. **Possible tuples.** Every tuple comes from the Cartesian product of the domains: $|D_1| \times |D_2| \times |D_3| = 3 \times 2 \times 4 =$ **24** possible tuples.
3. **Possible instances.** An instance is any subset of those 24 tuples. A set of 24 elements has 2^{24} subsets, so there are $2^{24} =$ **16,777,216** possible instances, counting the empty relation.
4. **Largest cardinality under the rule.** Each (roll_no, course) pair may appear only once. There are 3 × 2 = 6 pairs, so at most **6** tuples. This rule makes {roll_no, course} a key (Chapter 2.2).
5. **After the changes.** An update and an insert change only the instance; the schema is untouched. Degree stays 3. The update keeps the count, the insert adds one, so cardinality = 4 + 1 = **5**.

EXAM TRAPS

- A **relation** is a table; a **relationship** is an ER association between entities (an earlier lesson). Do not mix the words.
- Degree is the number of columns and cardinality the number of rows. Swapping them is the most common one-mark slip.
- An SQL table without a key can hold two identical rows (it is a multiset, or "bag"); a relation in the model cannot. Say this when asked how SQL differs from the model.
- Null is neither 0 nor the empty string ''. A student with cgpa null has an unknown cgpa, not a cgpa of zero.
- Count carefully: the number of possible tuples is a product of domain sizes; the number of possible instances is 2 raised to that product.

CHECK YOURSELF

- REMEMBER **Define degree and cardinality of a relation.** 2 marks

  Degree is the number of attributes (columns). Cardinality is the number of tuples (rows) in the current instance.

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

  E. F. (Ted) Codd of IBM, in 1970.

- UNDERSTAND **Why can a relation never contain two identical tuples?** 2 marks

  A relation is defined as a set of tuples (a subset of the Cartesian product of the domains), and a set holds each element at most once.

- UNDERSTAND **Distinguish a relation schema from a relation instance with an example.** 3 marks

  The schema is the design, such as Student(roll_no, name, dept, cgpa); it changes only through DDL. The instance is the set of tuples stored at one moment, such as the four rows of Figure 2.1.1; it changes with every insert, delete or update.

- APPLY **Three attributes have domains of sizes 5, 4 and 2. How many tuples and how many instances are possible?** 3 marks

  Tuples: 5 × 4 × 2 = **40**. Instances: every subset of 40 tuples, so $\mathbf{2^{40}}$ (about 1.1×10^{12}).

- APPLY **A relation of degree 5 and cardinality 200 gets one new column and 30 deleted rows. Give the new degree and cardinality, and say which change touched the schema.** 2 marks

  Degree 5 + 1 = **6**; cardinality 200 − 30 = **170**. Adding the column changed the schema; deleting rows changed only the instance.

- ANALYZE **A table Marks(roll_no, subjects) stores "DBMS, OS, CN" in one cell for each student. Which property of a relation does it break, and how would you fix it?** 4 marks

  It breaks atomicity: a cell holds a list, not one domain value, so the table is not in first normal form and a query such as "who takes OS?" must cut strings apart. Fix: store one tuple per (roll_no, subject) pair, for example (101, DBMS), (101, OS), (101, CN).

BEYOND THE SYLLABUS

Not for the exam. Real SQL engines use bag (multiset) semantics for speed and remove duplicates only when you write `DISTINCT`; search "set versus bag semantics SQL" to see where results differ. The same relational idea now runs at every scale: SQLite inside every Android phone, PostgreSQL and MySQL behind most web apps, and distributed SQL systems such as Google Spanner and CockroachDB spread one logical schema across data centres. The ACM/IEEE CS2023 curriculum places this material in its Data Management knowledge area. Codd's 1970 paper, "A Relational Model of Data for Large Shared Data Banks", is short and readable and is worth a look.

2.2 Keys: super, candidate, primary and foreign keys

A key is a set of attributes whose values pick out exactly one tuple, the way a roll number picks out one student. Keys stop duplicate facts and let one table point to another. Every later chapter uses them.

Four definitions, each built on the last

1. **Superkey.** A set K of attributes such that no two distinct tuples of any legal instance agree on all of K. In symbols: $t_1[K] = t_2[K] \Rightarrow t_1 = t_2$.
2. **Candidate key.** A superkey with no proper subset that is also a superkey, that is, a minimal superkey. A relation may have several.
3. **Primary key.** The one candidate key the designer chooses to identify tuples, underlined in the schema. The others become **alternate keys**. Prefer a key that is short, never null and never changes.
4. **Foreign key.** A set of attributes in a relation R_1 whose values must equal the primary key of some tuple in a relation R_2, or be null. R_1 is the referencing relation and R_2 the referenced one. R_2 may be R_1 itself, as when Employee.manager_id refers to Employee.emp_id.

A key is a property of the schema, decided from what the data means. One instance can prove a set is *not* a key (two rows agree on it) but can never prove that it is.

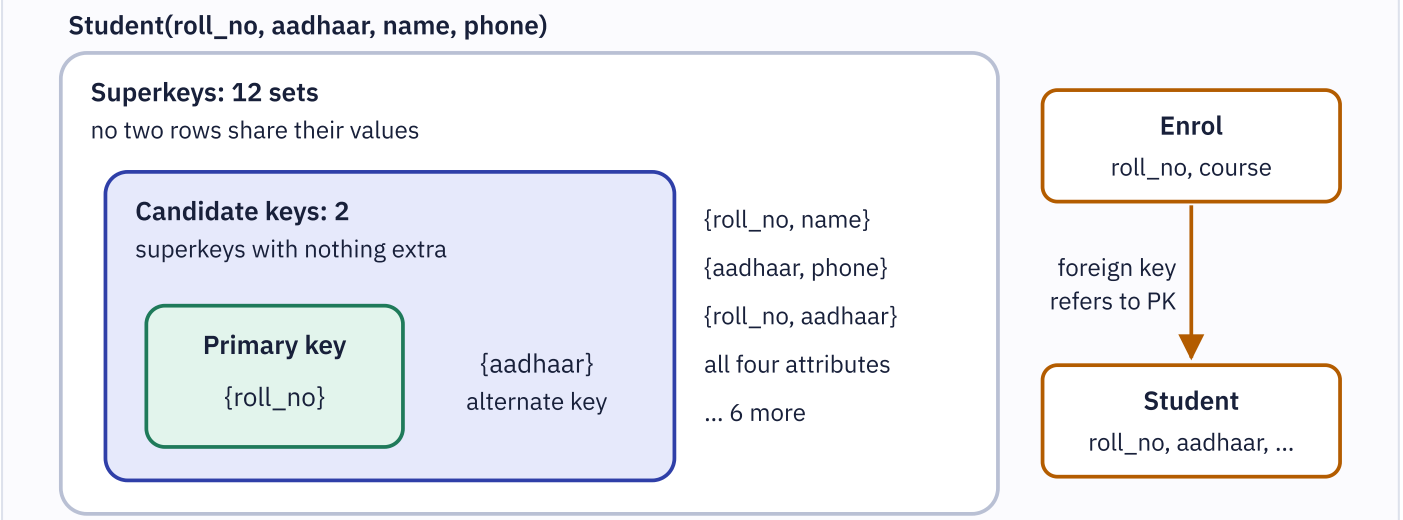


Figure 2.2.1 Keys nest: the primary key is one chosen candidate key, and every candidate key is a superkey. Notice that adding any attribute to a superkey gives another superkey, which is why there are 12. On the right, Enrol.roll_no is a foreign key that must match a Student primary key.

WORKED EXAMPLE

Question (5 marks). Student(roll_no, aadhaar, name, phone) has candidate keys {roll_no} and {aadhaar}. (a) How many superkeys does it have? (b) Enrol(roll_no, course, grade) has the single candidate key {roll_no, course}. How many superkeys? (c) Two rows of Student have name "Ravi". What does that tell you about {name}?

1. **Superkeys containing roll_no.** The other 3 attributes may each be in or out: $2^3 = 8$ sets.
2. **Superkeys containing aadhaar.** Likewise $2^3 = 8$ sets.
3. **Remove the double count.** Sets containing both keys were counted twice: the other 2 attributes are free, $2^2 = 4$ sets. Total = 8 + 8 − 4 = **12 superkeys**.
4. **Enrol.** Both roll_no and course must be present; only grade is free: 2^1 = **2 superkeys**, {roll_no, course} and {roll_no, course, grade}. In general, n attributes with one candidate key of k attributes give 2^{n-k} superkeys.
5. **The two Ravis.** Two distinct tuples agree on name, so {name} is **not a superkey**, and hence not a candidate key.

EXAM TRAPS

- "Minimal" means no attribute can be removed, not "fewest attributes". {roll_no} and a two-attribute key can both be candidate keys.
- Every candidate key is a superkey; most superkeys are not candidate keys.
- A primary key can never be null; a foreign key can be null unless declared NOT NULL.
- Do not "find" a key by looking at sample rows. Rows can only rule a set out.

CHECK YOURSELF

– REMEMBER **Define a candidate key.** 2 marks

A minimal superkey: a set of attributes that uniquely identifies tuples, with no proper subset that does.

– UNDERSTAND **Why is every candidate key a superkey, but not the other way round?** 2 marks

A candidate key is defined as a superkey that is also minimal. A superkey may carry extra attributes, such as {roll_no, name}, so it need not be minimal.

– APPLY **R(A, B, C, D, E) has one candidate key {A, B}. How many superkeys?** 2 marks

A and B fixed, C, D, E free: $2^{5-2} = 2^3 = \mathbf{8}$.

– APPLY **R(A, B, C) has candidate keys {A} and {B}. How many superkeys?** 2 marks

With A: $2^2 = 4$. With B: 4. With both: $2^1 = 2$. Total $4 + 4 - 2 = \mathbf{6}$.

– ANALYZE **Train(train_no, name, date, seats). Today no two rows share a name. Is {name} a candidate key? Is {train_no} enough if a train runs every day?** 4 marks

No to both. Today's rows cannot prove a key; two trains could share a name later. If one row is stored per train per date, train_no repeats every day, so the candidate key is {train_no, date}.

BEYOND THE SYLLABUS

Not for the exam. Many systems use a surrogate key, an auto-increment number or a UUID, instead of a natural key such as Aadhaar, partly because natural keys change and partly for privacy. Finding candidate keys from functional dependencies needs attribute closure; see a later lesson, Functional dependencies: Armstrong's axioms and closure.

Key terms of this lesson

TERM	MEANING OR FORMULA	CHAPTER
Relation	A subset of $D_1 \times D_2 \times \ldots \times D_n$: a set of tuples, no duplicates, no order	2.1
Degree / cardinality	Number of attributes / number of tuples	2.1
Possible tuples / instances	$\lvert D_1\rvert \times \ldots \times \lvert D_n\rvert = N$ tuples; 2^N instances	2.1
Schema / instance	The design $R(A_1, \ldots, A_n)$ / the set of tuples stored now	2.1
Superkey	$t_1[K] = t_2[K] \Rightarrow t_1 = t_2$ in every legal instance	2.2
Candidate / primary key	Minimal superkey / the candidate key chosen to identify tuples	2.2
Counting superkeys	One key of k attributes among n: 2^{n-k}; two one-attribute keys: $2^{n-1} + 2^{n-1} - 2^{n-2}$	2.2

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 **Q1. Explain the structure of the relational model: relation, tuple, attribute, domain, degree, cardinality, schema and instance. List the properties of a relation.** 10 marks

Each term with the Student example (Figure 2.1.1). Relation as a subset of the Cartesian product of domains. Properties: no duplicate tuples, no tuple order, no attribute order (named view), atomic values, null. Schema versus instance with the C declaration analogy; database schema and instance.

– REMEMBER **Q2. Define superkey, candidate key, primary key, alternate key and foreign key with examples, and show how they are related.** 8 marks

Formal definition of superkey; minimality for candidate keys; choice of primary key and alternate keys; foreign key with referencing and referenced relations, self-reference. Nested-set diagram (Figure 2.2.1). Keys come from meaning, not from one instance.

Apply

– APPLY **Q7. r(A, B) = {(1, x), (2, y), (3, z), (4, x)} and s(A, B) = {(2, y), (4, y), (5, z)}. Compute r ∪ s, r ∩ s, r − s, s − r, and the degree and cardinality of r × s.** 8 marks

r ∩ s = {(2, y)}, 1 tuple. r ∪ s = {(1, x), (2, y), (3, z), (4, x), (4, y), (5, z)}, 4 + 3 − 1 = 6 tuples. r − s = {(1, x), (3, z), (4, x)}, 3 tuples. s − r = {(4, y), (5, z)}, 2 tuples. r × s: degree 4, cardinality 4 × 3 = 12.

– APPLY **Q9. R(A, B, C, D, E) has candidate keys {A} and {B, C}. List the candidate keys, choose a primary key with a reason, and count the superkeys.** 8 marks

Superkeys containing A: $2^4 = 16$. Containing B and C: $2^3 = 8$. Containing A, B and C: $2^2 = 4$. Total 16 + 8 − 4 = **20**. Primary key {A}: one attribute, shorter to store in foreign keys. {B, C} becomes an alternate key, declared UNIQUE.

Analyze and evaluate

– ANALYZE **Q13. Design the keys and foreign keys for a train booking database with Train, Station, Passenger and Booking. Choose a referential action for each foreign key and justify it.** 13 marks

Spans Chapters 2.2 and 2.3. Candidate keys (train_no; station code; passenger id or Aadhaar; PNR), composite keys where a train runs on many dates. Booking references Train, Passenger and both Stations. RESTRICT on Train and Station deletes (history matters); CASCADE on PNR changes is rare; SET NULL only for optional links. Explain entity integrity for each key.

– ANALYZE **Q14. "SQL databases are relational." Evaluate this claim using Codd's rules and the formal definition of a relation.** 10 marks

Spans Chapters 2.1, 2.3 and 2.8. Meets: catalog tables, key access, one language, set-level updates, declared constraints, physical independence. Falls short: duplicate rows allowed (bags), three-valued null logic, limited view updating, bulk tools that skip checks, partial distribution independence. Conclusion: relational in practice, not fully by Codd's definition.

Glossary (terms used in this lesson)

Alternate key	A candidate key that was not chosen as the primary key.
Attribute	A named column of a relation, such as cgpa.
Candidate key	A minimal superkey: no attribute can be removed without losing uniqueness.
Cardinality	The number of tuples in a relation instance.
Cartesian product	r × s: every tuple of r combined with every tuple of s.
Degree	The number of attributes of a relation; also called arity.
Domain	The set of values an attribute may take.
Foreign key	Attributes of one relation whose values must match the primary key of another (or the same) relation, or be null.
Null	A marker for a missing or inapplicable value; not zero and not the empty string.
Primary key	The candidate key chosen to identify tuples; never null.
Relation	A set of tuples over a fixed list of attributes; a table in the relational model.
Schema	The design of a relation or database: names, attributes, domains and constraints.
Superkey	A set of attributes whose values are never shared by two distinct tuples.
Tuple	One row of a relation: one value for each attribute.

Sources and further reading

Textbooks

- Abraham Silberschatz, Henry F. Korth and S. Sudarshan, *Database System Concepts*, 7th ed., McGraw-Hill. Chapter 2, "Introduction to the Relational Model": relations, schemas, keys, relational algebra. The calculi are in the chapter on formal relational query languages.
- Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems*, 7th ed., Pearson. Chapters on the relational data model and relational database constraints, and on the relational algebra and relational calculus.
- Raghu Ramakrishnan and Johannes Gehrke, *Database Management Systems*, 3rd ed., McGraw-Hill. Chapters on the relational model and on relational algebra and calculus.
- C. J. Date, *An Introduction to Database Systems*, 8th ed., Pearson/Addison-Wesley. Relations, keys, integrity and the relational algebra and calculus.
- E. F. Codd, "A Relational Model of Data for Large Shared Data Banks", *Communications of the ACM*, 1970. The paper that started the relational model.

Web resources

The writing session's network could not open web pages, so none of these was opened while writing. They are well-known open resources, listed from memory:

- Wikipedia, "Relational algebra", "Tuple relational calculus" and "Domain relational calculus"
- Wikipedia, "Codd's 12 rules"
- PostgreSQL documentation, chapter "Constraints" (foreign keys and referential actions): postgresql.org/docs/current/ddl-constraints.html

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.