Database Management Systems

Domain calculus and Codd's rules

The notes for the lesson “Domain calculus and Codd's rules”: chapters 2.7 to 2.8, with their figures, worked examples, exam traps, quizzes and exam question bank, in one file.

2 chapters | Chapters 2.7 to 2.8

About these notes

These are the notes for the University free n easy video lesson **Domain calculus and Codd's rules** on Database Management Systems: chapters 2.7 to 2.8, with every figure, worked example, exam trap and quiz, and the long-answer exam questions on these chapters.

2.7 Domain relational calculus

Domain relational calculus (DRC) is the second form of relational calculus. Its variables stand for single attribute values, members of a domain, instead of whole tuples. A query has the form $\{\langle x_1, x_2, \ldots, x_n\rangle \mid P(x_1, x_2, \ldots, x_n)\}$: the set of value lists that make P true. QBE (Query-By-Example), designed at IBM by Moshé Zloof in the 1970s, is built on it, and so are the fill-in query grids of desktop tools.

You need: Chapter 2.6, Tuple relational calculus; this chapter only shows what changes.

Domain variables and atoms

The connectives ∧, ∨, ¬, ⇒ and the quantifiers ∃ and ∀ work exactly as in TRC. Only the atoms change:

1. **$\langle x_1, \ldots, x_n\rangle \in r$**: the values x_1 to x_n, in the order of r's attributes, form a tuple of r. For Student(roll_no, name, dept, cgpa), ⟨r, n, d, g⟩ ∈ Student binds r to roll_no, n to name, d to dept and g to cgpa. A constant may stand in a position, as in ⟨r, n, 'CSE', g⟩.
2. **x θ y** and **x θ c**: compare two domain variables, or a variable and a constant.

Every variable that is not in the result list must be bound by ∃ (written ∃r, d, g for ∃r ∃d ∃g). A join needs no equality: using the **same variable** in two atoms forces the values to be equal.

Safety and power are as in TRC: safe DRC expresses exactly what relational algebra does.

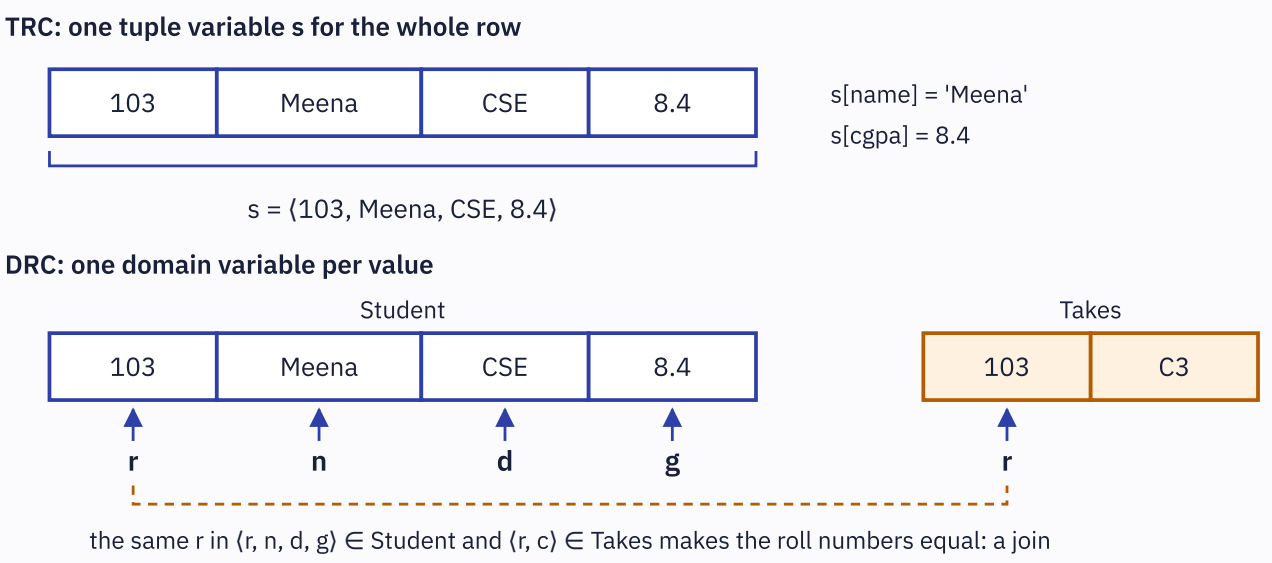


Figure 2.7.1 The same Student row seen by the two calculi. TRC names the whole row s and reaches values as s[name]; DRC names each value directly, in attribute order. Notice the shared variable r: repeating it across Student and Takes is how DRC writes a join.

WORKED EXAMPLE

Question (6 marks). Using Student of Chapter 2.4 and Takes of Chapter 2.6, write DRC for (a) the name and cgpa of every ECE student; (b) the names of students who took C3. Evaluate both.

1. **(a) Formula.** The result needs n and g; r and d must be bound: {⟨n, g⟩ | ∃r, d (⟨r, n, d, g⟩ ∈ Student ∧ d = 'ECE')}. Putting the constant in its position gives the shorter {⟨n, g⟩ | ∃r (⟨r, n, 'ECE', g⟩ ∈ Student)}.
2. **(a) Evaluate.** The ECE rows are 102 and 106: **{(Ravi, 7.8), (Karthik, 8.8)}**, 2 tuples of degree 2.
3. **(b) Formula.** {⟨n⟩ | ∃r, d, g (⟨r, n, d, g⟩ ∈ Student ∧ ⟨r, 'C3'⟩ ∈ Takes)}. The shared r links each student to their own Takes rows.
4. **(b) Evaluate.** Takes rows with C3 have roll numbers 101, 103 and 105. Their names: **{Asha, Meena, Priya}**, 3 tuples.

EXAM TRAPS

- Positions follow the attribute order. ⟨n, r, d, g⟩ ∈ Student would bind n to roll_no.
- Each atom ⟨...⟩ ∈ r needs one variable or constant for every attribute of r, even ones you do not use.
- Forgetting ∃ on r, d or g leaves free variables that are not in the result list: an invalid query.
- DRC has nothing to do with the "domain constraint" of Chapter 2.3, though both use the word domain.

CHECK YOURSELF

– REMEMBER **Write the general form of a DRC query.** 1 mark

{⟨x_1, x_2, ..., x_n⟩ | P(x_1, x_2, ..., x_n)}, where the x_i are domain variables and P is a formula.

– UNDERSTAND **How do the variables of DRC differ from those of TRC?** 2 marks

A TRC variable ranges over whole tuples of a relation; a DRC variable ranges over single values of an attribute's domain.

– APPLY **Write DRC for names of students who took no course, and evaluate it on this unit's data.** 3 marks

{⟨n⟩ | ∃r, d, g (⟨r, n, d, g⟩ ∈ Student ∧ ¬∃c (⟨r, c⟩ ∈ Takes))}. Roll numbers 104 and 106 have no Takes rows: **{John, Karthik}**.

– APPLY **Write DRC for roll numbers with cgpa at least 8.4, and evaluate it.** 2 marks

{⟨r⟩ | ∃n, d, g (⟨r, n, d, g⟩ ∈ Student ∧ g ≥ 8.4)}. 9.1, 8.4 and 8.8 qualify: **{101, 103, 106}**.

– ANALYZE **Why do visual query grids such as QBE follow domain calculus rather than tuple calculus?** 3 marks

A grid shows one column per attribute, and the user types a value or a variable into a cell. Each cell is exactly a DRC position, and typing the same variable in two tables' grids is a DRC join by a shared variable.

BEYOND THE SYLLABUS

Not for the exam. Datalog, a rule language used in program analysis and in some data systems, is positional like DRC: answer(N) :- student(R, N, 'CSE', G), G > 8. It adds recursion, which neither calculus has. Search "Datalog" and "Query-By-Example Zloof".

2.8 Codd's rules for relational databases

By the 1980s many products called themselves "relational" while keeping their old pointer-based designs underneath. In 1985 Codd answered with a list of rules, published in two Computerworld articles, that a system must follow to deserve the name. They are called the twelve rules, but they are numbered 0 to 12, so there are thirteen. Together they protect the model's two promises: all data is reached through values in tables by one high-level language, and programs keep working when storage or structure changes.

You need: Chapter 2.1 and Chapter 2.3; also an earlier lesson, Views of data and three-schema architecture, for rules 8 and 9.

The thirteen rules

NO.	RULE	WHAT IT DEMANDS
0	Foundation rule	The system must manage its databases entirely through its relational capabilities.
1	Information rule	All information, including table and column names, is stored as values in tables.
2	Guaranteed access	Every single value can be reached by table name + primary key value + column name, with no pointers or positions.
3	Systematic treatment of nulls	A null marks missing or inapplicable information, is distinct from zero and the empty string, and works the same for every data type.
4	Dynamic online catalog	The description of the database (the catalog or data dictionary) is itself stored as tables and queried with the same language.
5	Comprehensive data sublanguage	At least one language with a plain text syntax supports data definition, view definition, data manipulation, integrity constraints, authorisation and transaction boundaries.

NO.	RULE	WHAT IT DEMANDS
6	View updating	Every view that is theoretically updatable can be updated by the system.
7	High-level insert, update and delete	Insert, update and delete work on whole sets of rows in one statement, not just one row at a time.
8	Physical data independence	Programs are unaffected by changes in storage or access methods.
9	Logical data independence	Programs are unaffected by changes to tables that keep the same information, such as splitting a table.
10	Integrity independence	Integrity constraints are defined in the data sublanguage and stored in the catalog, not coded in programs.
11	Distribution independence	Programs work the same whether the data sits on one machine or is spread over many.
12	Non-subversion rule	If a low-level, record-at-a-time interface exists, it cannot be used to bypass the integrity rules.

Five groups that make them easy to remember

Learning thirteen rules as a list is hard; learning five ideas is easy. Rule 0 is the **foundation** that the rest spell out. Rules 1 to 4 fix the **structure**: data, nulls and even the catalog are values in tables. Rules 5 to 7 fix the **language**: one language does everything, including updates on views and on whole sets of rows. Rules 8 to 11 promise **independence**: programs survive changes to storage, table structure, constraints and location. Rule 12 is **protection**: there is no back door around the rules.

Rules 8 and 9 are the three-schema architecture of an earlier lesson written as a test. Physical independence (rule 8) lets the DBA add an index or move files without touching programs. Logical independence (rule 9) lets the DBA split a table and keep old programs running through a view with the old table's name.

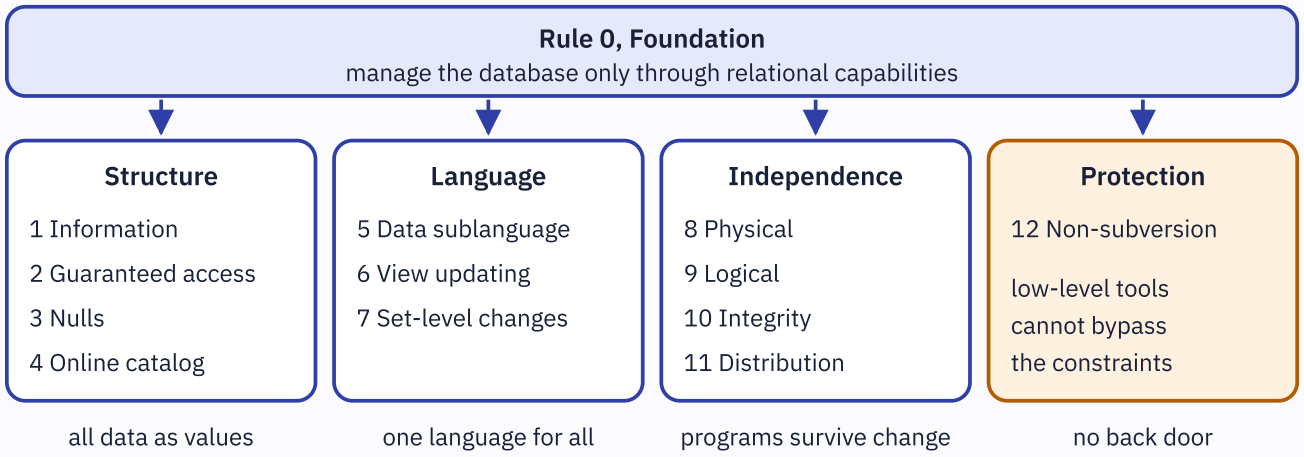


Figure 2.8.1 Codd's thirteen rules as one foundation and four groups. Read the bottom line of each group as the promise it makes. Notice that the independence group is the largest: most of the rules exist so that programs keep working while the database changes underneath them.

WORKED EXAMPLE

Question (8 marks). Student is the six-row table of Chapter 2.4, with one more row (107, Divya, ECE, null) for a student whose cgpa is not yet known. For each statement or event below, name the Codd rule it shows and give the result.

```
(a) SELECT column_name FROM information_schema.columns
    WHERE table_name = 'student';
(b) SELECT cgpa FROM Student WHERE roll_no = 103;
(c) UPDATE Student SET cgpa = cgpa + 0.1 WHERE dept = 'CSE';
(d) CREATE INDEX idx_dept ON Student(dept);   -- then rerun (c)
(e) SELECT name FROM Student WHERE cgpa IS NULL;
(f) ALTER TABLE Student ADD CHECK (cgpa BETWEEN 0 AND 10);
    UPDATE Student SET cgpa = 10.5 WHERE roll_no = 101;
```

1. **(a) Rule 4, online catalog.** The table's own description is read with ordinary SQL from catalog tables. Result: 4 rows, **roll_no, name, dept, cgpa**.
2. **(b) Rule 2, guaranteed access.** Table name Student, key value 103 and column cgpa reach one value: **8.4**.
3. **(c) Rule 7, set-level update.** One statement changes every CSE row: 101 from 9.1 to **9.2**, 103 from 8.4 to **8.5**, 105 from 7.5 to **7.6**. **3 rows** updated.
4. **(d) Rule 8, physical independence.** The index changes how rows are found, not what the statement means. Statement (c) needs no change and still updates the same 3 rows; only its speed may change.
5. **(e) Rule 3, nulls.** Result: **Divya**, 1 row. Writing `WHERE cgpa = NULL` instead returns **0 rows**, because a comparison with null is "unknown", never true.
6. **(f) Rule 10, integrity independence.** The constraint lives in the catalog, not in any program. The update to 10.5 breaks it, so it is **rejected**, whichever program sends it, and Asha's row keeps its old cgpa.

EXAM TRAPS

- "Twelve rules" are numbered 0 to 12: there are thirteen. Write rule 0 too.
- Rule 1 says data is stored as values in tables; rule 2 says each value can be reached by table, key and column. Do not merge them.
- Rule 8 (physical: indexes, files) and rule 9 (logical: table structure) are the pair examiners most often swap.
- Rule 10 puts constraints in the catalog; rule 12 stops low-level tools from going around them.
- No commercial product satisfies all thirteen rules. Do not write that Oracle or MySQL "follows all twelve". And Codd's rules (1985) are not Codd's theorem (1972) of Chapter 2.6.

CHECK YOURSELF

– REMEMBER **State the information rule.** 1 mark

All information in a relational database, including table and column names, is represented explicitly as values in tables.

– REMEMBER **When and why did Codd publish his rules, and how many are there?** 2 marks

In 1985, to test whether products sold as relational really were. They are numbered 0 to 12, so thirteen rules.

– UNDERSTAND **Distinguish physical from logical data independence with one example each.** 4 marks

Physical (rule 8): adding an index or moving data files does not change programs. Logical (rule 9): splitting Student into two tables and creating a view called Student keeps old programs running.

– UNDERSTAND **Why should the catalog be stored as relational tables?** 2 marks

So that users and tools can find out what tables, columns and constraints exist with the same query language they use for data, with no special interface.

– APPLY **One UPDATE statement adds 5% interest to the 1,200 savings accounts among 5,000 accounts. Which rule does it use, and how many rows change?** 2 marks

Rule 7, high-level (set-level) update. **1,200** rows change in one statement.

– APPLY **A bulk-load utility writes rows straight into data files and skips the foreign key checks. Which rule is broken?** 2 marks

Rule 12, non-subversion: a low-level interface is bypassing the integrity rules that the high-level language enforces.

– ANALYZE **Why do most SQL products only partly meet rule 6?** 4 marks

Updating a view means turning the change into changes on the base tables. For a view with joins, aggregates or DISTINCT there may be many possible translations or none. Deciding this in general is hard, so products allow updates only on simple views (usually one table, no grouping) or through triggers written by hand.

BEYOND THE SYLLABUS

Not for the exam. C. J. Date and Hugh Darwen's *The Third Manifesto* argues that SQL itself breaks the model, through duplicate rows and nulls, and proposes a stricter language. NoSQL databases drop several rules on purpose, such as a fixed schema and joins, to scale out cheaply (a later lesson, NoSQL databases: document, key-value and graph). NewSQL and distributed SQL systems try to meet rule 11 at internet scale. Search "The Third Manifesto" and "NewSQL".

Key terms of this lesson

TERM	MEANING OR FORMULA	CHAPTER
TRC / DRC query	$\{t \mid P(t)\}$ / $\{(x_1, \ldots, x_n) \mid P(x_1, \ldots, x_n)\}$	2.6, 2.7
Codd's rules (1985)	13 rules numbered 0 to 12: foundation, structure, language, independence, protection	2.8

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

- REMEMBER **Q4. State and explain Codd's rules for relational database systems.** 16 marks

  Why Codd wrote them (1985). All thirteen rules, 0 to 12, each with one line of meaning and an SQL example where possible (catalog query, key access, IS NULL, set-level UPDATE, index for physical independence, CHECK for integrity independence). Grouping into foundation, structure, language, independence, protection (Figure 2.8.1). No product satisfies all.

Apply

- APPLY **Q8. Write the queries of Q6 in tuple relational calculus and in domain relational calculus, and evaluate the "every CSE course" query.** 12 marks

  Patterns of Chapter 2.6 (selection, projection, join, negation) and their DRC forms with positional variables and shared variables for joins (Chapter 2.7). For all: ∀c ∈ Course (c[dept] = 'CSE' ⇒ ∃k ∈ Takes (...)). Evaluation grid (Figure 2.6.1): {Asha, Priya}. State safety.

Analyze and evaluate

- 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)

Attribute	A named column of a relation, such as cgpa.
Catalog	The tables that describe the database itself: its tables, columns, constraints and users. Also called the data dictionary.
Codd's theorem	Safe tuple calculus, safe domain calculus and relational algebra have equal expressive power.
Degree	The number of attributes of a relation; also called arity.
Domain	The set of values an attribute may take.

Domain relational calculus	A declarative query language whose variables range over attribute values.
Foreign key	Attributes of one relation whose values must match the primary key of another (or the same) relation, or be null.
Free variable	A variable in a formula not bound by any quantifier; in a query only the result variable may be free.
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.
Relational algebra	A procedural query language of operators that take relations and return relations.
Schema	The design of a relation or database: names, attributes, domains and constraints.
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.