Relational Model

↳ Data is stored as table of values, a flat file with records.

→ A row (record) describes an entity/association among entities.

→ Tablename and Column name helps to interpret the data.

→ table: RELATION
row: TUPLE
column: ATTRIBUTE

→ Each attribute has a domain (is a set of ATOMIC values); i.e., the elements of a domain can't be broken down any further, is a final data, with respect to the requirements of an end application.
Domains can be formally specified with data type and/or size.
Attributes can only take values from its domain.

↳ Relation schema of a relation R: (Also known as INTENSION OF A RELATION)

→ $R(A_1, A_2, \dots A_n)$ is a relation schema of degree n (# of attributes) where A_i is an attribute ($\forall i$) with D_i = Domain(A_i).

↳ Relation / state-of-a-relation of a schema $R(A_1, A_2, \dots A_n)$:

→ $r(R)$ = set of n-tuples $\{t_1, t_2, \dots t_m\}$ where t_i is a ordered list of values, $t_i = \langle v_1, v_2, \dots v_n \rangle$, where $v_j \in$ domain(A_j) $\forall j \in [n]$.
Here, m is the # of entries to the relation, n is the degree of the schema

→ $r(R) \subseteq Dom(A_1) \times Dom(A_2) \times \dots \times Dom(A_n)$

↳ Characteristics of Relations:

→ Tuples are not assumed to be stored in any sort of order.

→ Attributes in a tuple maybe ordered or unordered. (NOTE: above defn assumes ordered)
For unordered, every tuple t_i is a set of attribute-value pairs.

→ Values and Null values: Attributes are atomic valued; and for a tuple, singular. Attribute values are either from its domain, or NULL (absence of data).
(This is known as FIRST NORMAL FORM)
[interpretation of why a value is NULL is difficult, many reasons might be possible]

→ Meaning of a relation: Can be thought of as a type declaration.

→ No two tuples are same, all records are unique. (constraint).

↳ Constraints of Relational Model:

① Model-Based: Inherent in the model. (tuple uniqueness, no ordering of tuples, etc).

② Schema-Based: Can be defined at the time of schema definition using DDL statements

③ Application-Based: Logical restrictions that depends on application working on it.

④ Data dependancy constraints: (i) Functional dependancy, (ii) multivalued dependancy.

→ Schema Based Constraints:

▷ Domain Constraint: Values of attributes belongs to its domain. 2

▷ Key Constraint: if SK is an attribute of schema R, such that $\forall i,j \in [m]$, $t_i[SK] \neq t_j[SK]$, i.e., the values list of these attr.s are unique ∀ tuples, then SK is called a SUPERKEY.
(Note that entire Attr. set is a superkey).
The minimal superkey (that has no superkey proper subset) is called a candidate key. (There can be multiple such).
One of the candidate keys are chosen by the designer as the primary key, which is preferably the smallest candidate key.
Preferences of primary key:
→ smaller size
→ Alphanumeric/Numeric over purely alphabetical.
→ In the context of application, the mostly used identifying candidate key is used as the primary key. (Eg roll no.)

▷ Key Integrity: Primary key is not NULL (used for identifying a tuple).

↳ Relational Database Schema:

→ Set of individual relational schemas $\{R_1, R_2, \dots R_K\}$, and set of integrity constraints.

↳ Relational Database State:

→ $DB = \{r_1(R_1), r_2(R_2), \dots r_K(R_K)\}$ → The set of states of the relations.

↳ Integrity Criterias:

→ Key Integrity: Primary key is not NULL.

→ Referential Integrity: A subset of attributes "FK" of a relation R1, references a relation R2, if the domain of FK in R1 is same as the primary key of R2 (PK).
~~[illegible]~~
▷ For each tuple $t_1 \in r_1(R_1)$, $\exists\, t_2 \in r_2(R_2) \ni t_1[FK] = t_2[PK]$, or $t_1[FK] == NULL$
▷ R_1 is the referencing relation, R_2 is the referenced relation.
▷ FK is the foreign key of R_1 that references the relation R_2.

OPERATION	REFERENCING RELATION	REFERENCED RELATION
INSERTING A TUPLE	Allowed only if value of FK of new tuple being added is present in referenced rel^n.	Only uniqueness of PK is to be ensured.
MODIFYING A TUPLE	If FK is changed, it needs to be present in the referenced relation.	If PK is changed, allow only if the old one wasn't referenced.
DELETING A TUPLE	No issue, allowed.	If PK is currently referenced, not allowed; else no issue.

↳

Relational Model

offers language to access data from database

Relational Algebra

- → Expressions specifies sequence of OPERATIONS to be applied.
- → input: multiple relations
  output: another relation
- → Procedural
- → Provides foundations of operations in relational models.
- → Internally used for implementing and optimizing queries.
- → In commercial SQL also, some concepts are utilized.

Relational Calculus

- → Specifies what data to needed, not how to get it.
- → Non-Procedural.
- → Based on mathematical logic.
- → Some concepts are utilized in SQL.

↳ **Relational Algebra:**

→ **Select Operation:**

- ▷ $\sigma_{PREDICATE}(RELATION)$.
- ▷ output relation schema is same as input relation.
- ▷ Outputs those records that satisfy the predicate.
- ▷ Eg: $\sigma_{ROLL=1}(STUDENT)$, $\sigma_{DCODE="CSE" \&\& SCORE>75}(STUDENT)$
- ▷ "&&" predicates can be cascaded for optimization; and is commutat

$$\sigma_{A_1=v_1,\, A_2=v_2}(R) \equiv \sigma_{A_1=v_1}\left(\sigma_{A_2=v_2}(R)\right).$$

→ **Project Operation:**

- ▷ $\Pi_{A_1, A_2 \cdots A_K}(R)$
- ▷ Output schema is as specified by the operation expression.
- ▷ outputs all records' specified attributes.
- ▷ This is not commutative (non cascadeable).

→ **Rename / Assignment Operation:**

- ▷ $\rho[S(B_1, B_2, \dots B_n)](R)$ → renames R to S & A_i to B_i $\forall i \in [n]$.
- ▷ $\rho[S](R)$ → renames R to S.
- ▷ $\rho[(B_1, B_2, \dots B_n)](R)$ → renames A_i to B_i $\forall i \in [n]$

→ **Cartesian Product:**

- ▷ $R_1 \times R_2$ → $(R_1.A_1, R_1.A_2, \dots R_1.A_m, R_2.B_1, R_2.B_2, \dots R_2.B_n)$
- ▷ degree $= m+n$, # of tuples $= K_1 \times K_2$.
- ▷ may result in meaningless records; i.e. when same attributes contradict.

→ **Equijoin / Θ-join:** (If predicate has only '==', then equijoin, if there other comp ⇒ Θ-join)

- ▷ $R_1 \bowtie_{PREDICATE} R_2 \equiv \sigma_{PREDICATE}(R_1 \times R_2)$ → fixes meaningless joins.

→ <u>Natural join:</u>

- ▷ compares equality of common attributes, and joins only those agreeing records.
- ▷ same attributes in different relations are not repeated, are shown only once.
- ▷ $R_1 * R_2$ → natural join denotion.
- ▷ Sometimes we need to rename some attributes before natural join.

→ <u>Set operations:</u>

- ▷ Performed on union compatible relations:
  - → degree must be same of both relations
  - → $\text{Domain}(A_i) = \text{Domain}(B_i) \ \forall\, i \in [n]$.
- ▷ The resultant relation's attribute names can be dependent on implementation, hence, renaming is a good practise.
- ▷ $R_1 \cup R_2$, $R_1 - R_2$, $R_1 \cap R_2$ are some set operations.
- ▷ The tuples are treated as the elements, as a whole.