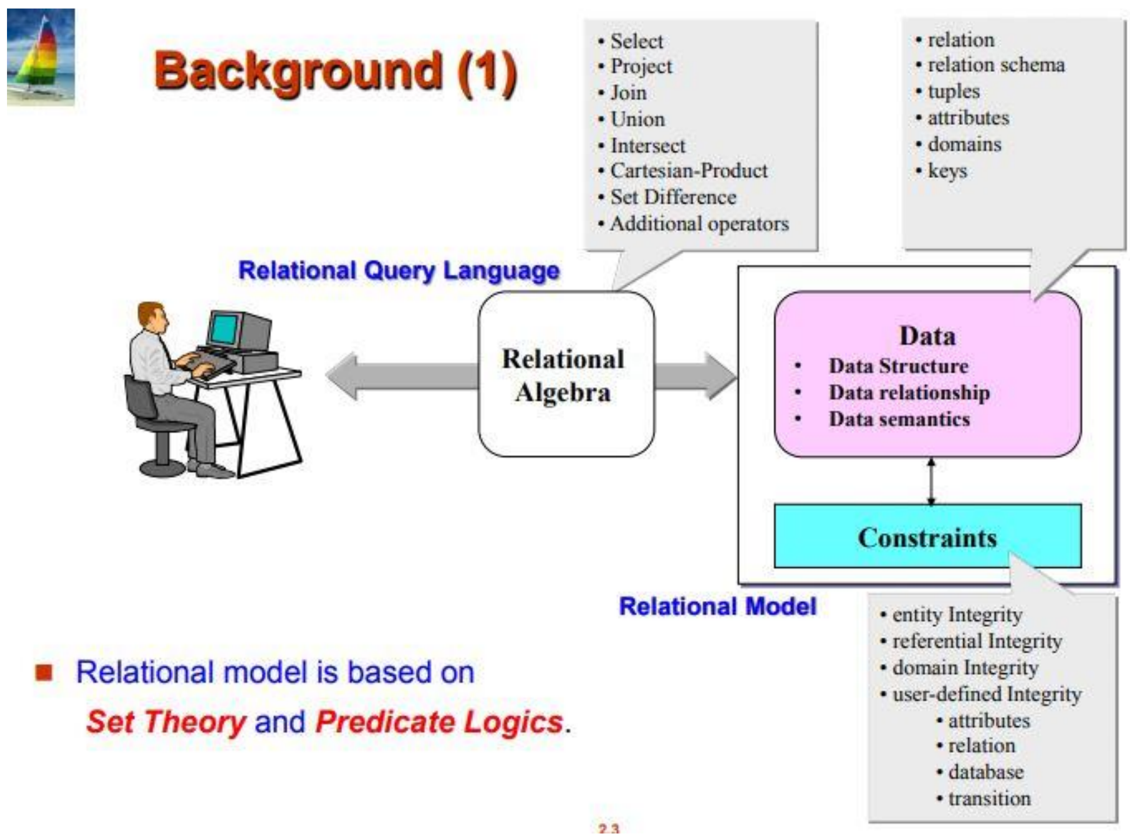
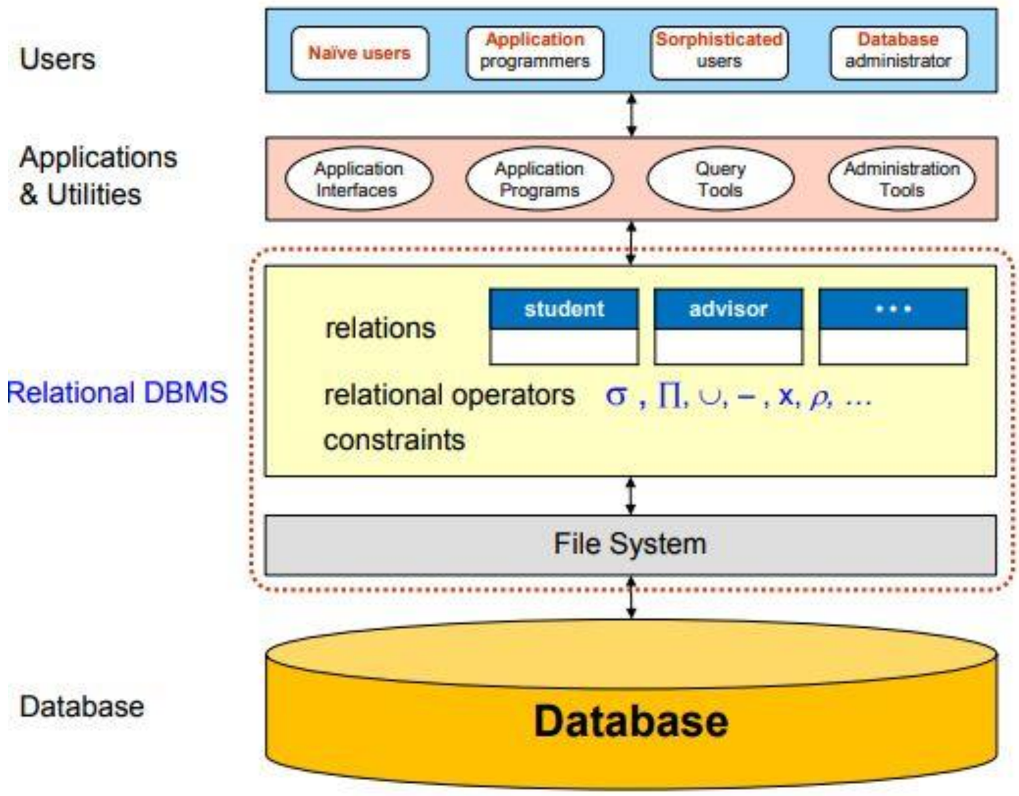
Chapter 2. Introduction to the Relational Model





2.1 Structure of Relational Databases

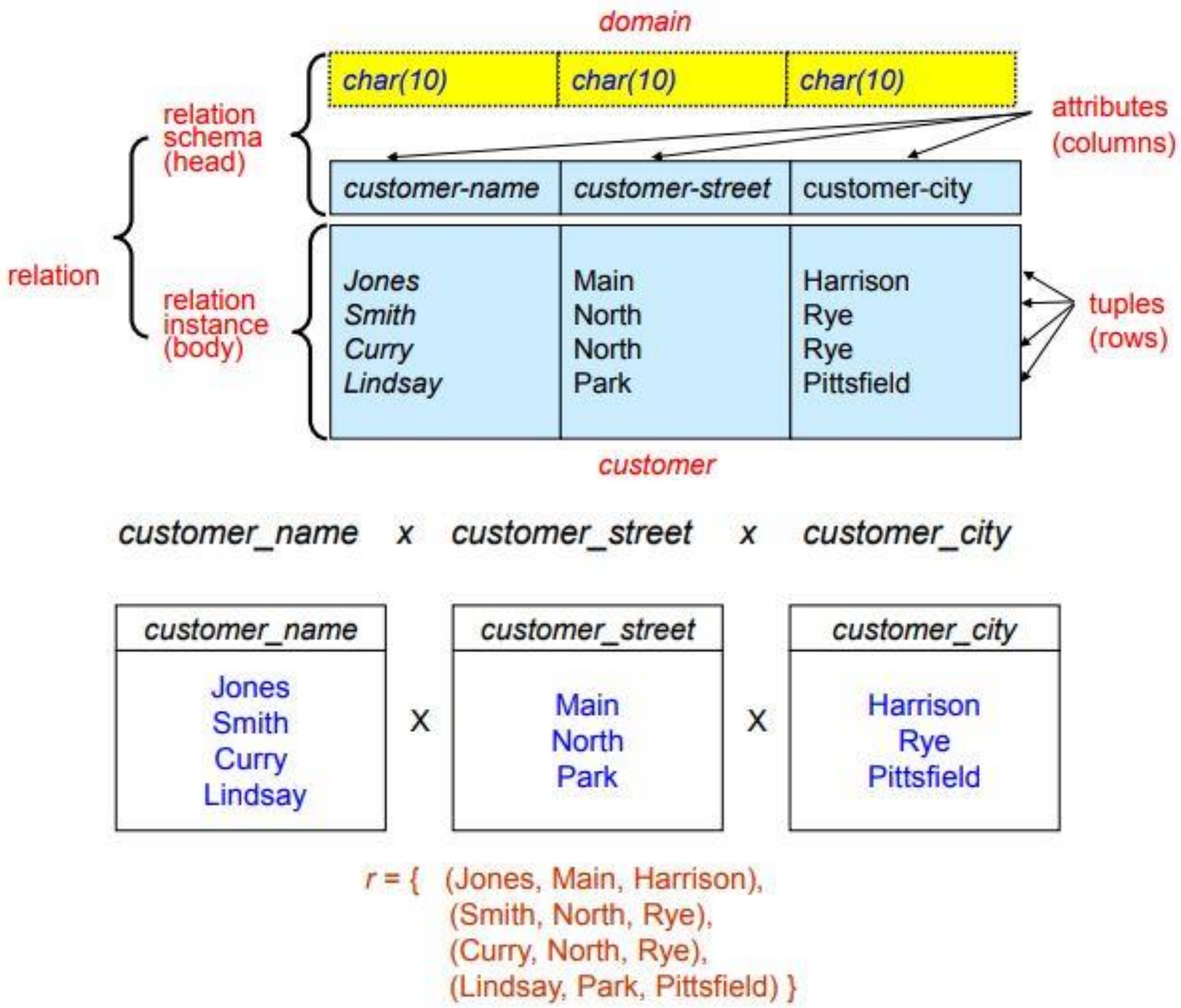
A relational database consists of a collection of tables, each of which is assigned a unique name. Relation is a mathematical concept explaind below.

Given sets $D_1, D_2, \ldots, D_n$, a relation r is a subset of $D_1 \times D_2 \times \ldots \times D_n$

Thus, a relation is a set of n-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$ which matches to a table. Each tuple is a row in a table. And for the column of a table, we use the term attribute. For each attribute of a relation, there is a set of permitted values, called the domain of that attribute.

As a relation is a set, the order of tuples is irrelevant, but also the order of attributes is irrelevant. However, for the domain, we require it be atomic, which means all attribute values must not be subdivided. Multivalued attribute values or composite attribute values are not allowed.

The null value signifies that the value is unknown or does not exist.



2.2 Database Schema

Database schema should be differentiated with database instance. **Database schema** is the logical design of the database, and **database instance** refers to a snapshot of the data at a given instant in time.

$A_1, A_2, \ldots, A_n$ are *attributes*

$R = (A_1, A_2, \ldots, A_n)$ / $R = (A_1{:}D_1, A_2{:}D_2, \ldots, A_n{:}D_n)$ **is a *relation schema***

ex) *Department_schema = (dept_name, building, budget)*

***r(R)* is a *relation* on the relation schema R**

ex) *department (Department_schema)*

Bad design of relations result in **repetition of information**, **the need for null values**.

2.3 Keys

Every tuple must be unique. The way to distinguish tuples uniquely is by attributes.

$K \subseteq R$(Relation schema)

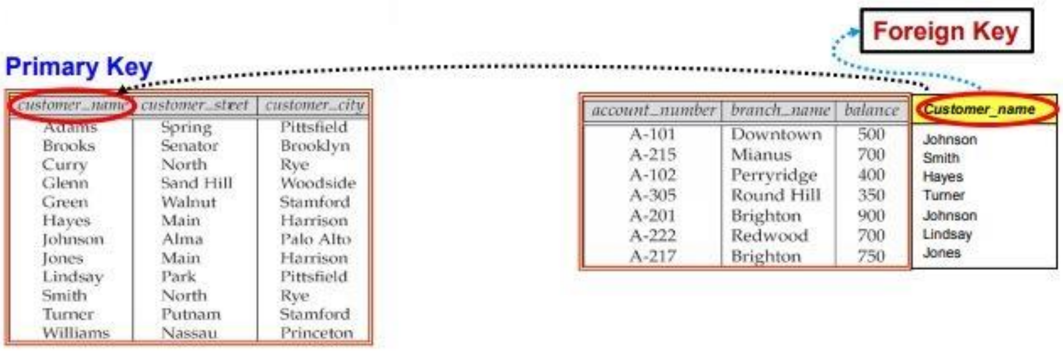
K is a **superkey** if values for ***K*** allow us to identify uniquely a tuple in the relation ***r(R)***. That is, if 2 tuples t_1 and t_2 are in ***r(R)*** and $t_1 \neq t_2$, then $t_1.K \neq t_2.K$.

ex) {*dept_name*}, {*dept_name, building*} are both superkeys of *department*

If ***K*** is a superkey, any superset of ***K*** is also a superkey. However, when ***K*** is minimal we call this ***K*** a **candidate key**.

ex) {*dept_name*} is a candidate key for *department*.

A **primary key** is a candidate key chosen by the database designer as the principal means of identifying tuples within a relation.



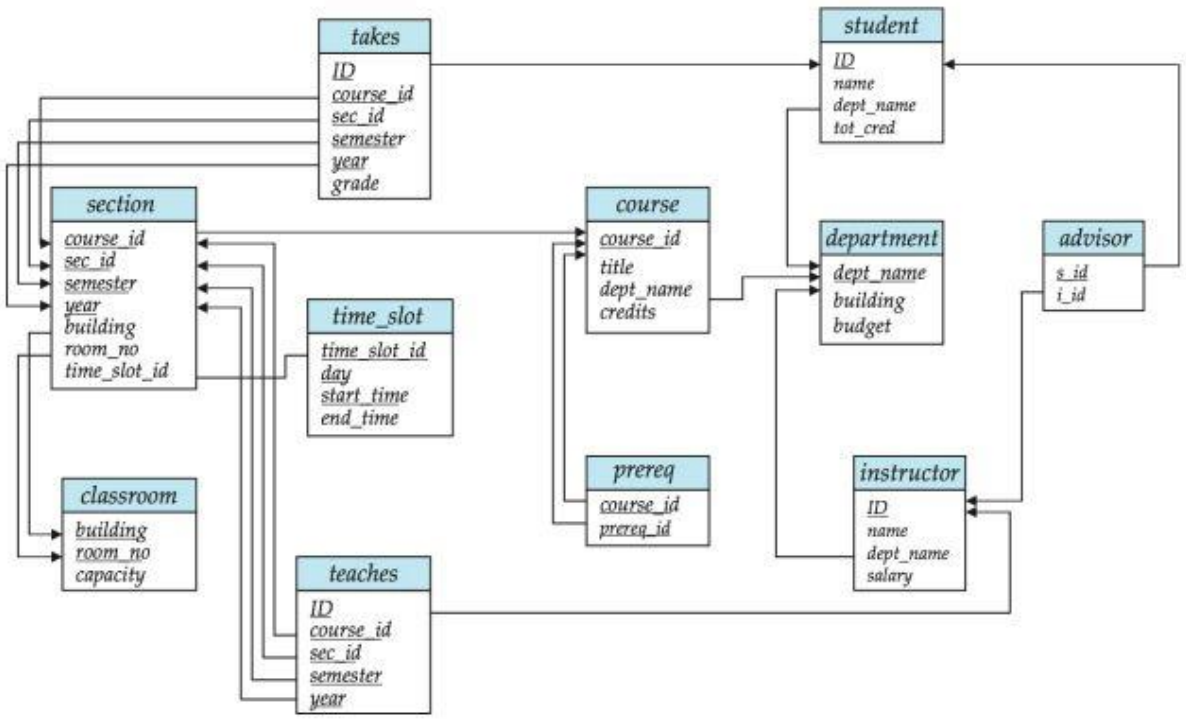
A relation r_1(*account* in this case) may include among its attributes the primary key of another relation r_2(customer). This attribute(*customer_name*) is called a **foreign key**. The relation r_1 is called the **referencing relation** of the foreign key dependency, and r_2 is called the **referenced relation**.

There are a few Integrity constraints that a relational model needs to satisfy.

- **Entity Integrity**: no primary key value can be null. If there could exist some tuples whose all attribute values are the same, make artificial attribute to use as a primary key.
- **Referential Integrity Constraint**: all foreign key values in a referencing relation should appear only once in the referenced relation as a primary key.
- **Domain Integrity**: every element from a relation should respect the type and restrictions of its corresponding attribute.
- **User Defined Integrity**: to assert business structure.

2.4 Schema Diagrams

Schema Diagram for University database



2.5 Relational Query Languages

3 types

- Relational algebra -> Procedural
- Tuple relational calculus -> Non-procedural
- Domain relational calculus -> Non-procedural

2.6 Relational Operations

In later chapters… Here are some operators.

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{salary>=85000}(instructor)$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID,\ salary}(instructor)$
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
⋈ (Natural Join)	$instructor \bowtie department$
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
× (Cartesian Product)	$instructor \times department$
	Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes)
∪ (Union)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the two input relations.