# The Relational Paradigm

Without a Language or Syntax!

*What is the relational world all about?*

## *The Relational Ontology*

This ontology is consistent with the practice in computer science and information science categorizing a domain of concepts (i.e. individuals, attributes, classes and relationships). This ontology of the relational paradigm of data modeling minimizes the vestiges of implementation languages and methodologies to expose the core nature of relational concepts.

### 1. Individuals

The most concrete concept in the relational paradigm is the ***tuple***.

#### 1.1. Tuple

A *tuple* corresponds 1-1 with a single concept of reality that it represents. A *tuple* collects the facts that identify it as a single concept and the facts most closely identified with it.

### 2. Attributes

*Attributes* are those characteristics (facts) that describe a *tuple*. In the relational paradigm *attributes* define data characteristics - each of which has a static and dynamic form. A prescribed set of *attributes* defines what is called the ***structure*** of a *tuple*. From inception to extinction the *structure* of a *tuple* is immutable. The number of *attributes* in a *tuple* is called its ***degree***.

#### 2.1. Data Attribute

*Data attributes* store information (data) in the *tuple* and implement the property of ***remembrance***. *Remembrance* is manifest in each *attribute* dynamically as “what *is* remembered,” a particular ***data attribute value*** for each *tuple* derived from a ***data attribute domain*** that statically defines “what *can* be remembered,” the possible values of the *attribute*.

### 3. Classes

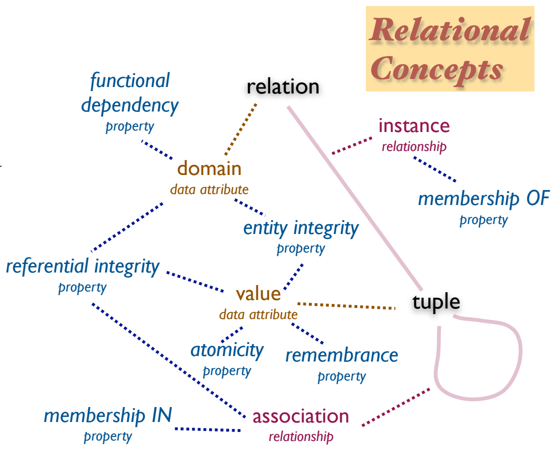
The relational paradigm groups individuals into a collection called a ***relation***. The *relation* corresponds directly with its mathematical antecedent where *attribute* values within each *tuple* reflect a correspondence with the coincidence of facts in the “real world,” a correspondence (*attribute* relationship) that is shared by every *tuple* in that *relation*.

#### 3.1. Relation

The ***relation*** concept combines both a definition of *structure* and the collection of *tuple(s)* based on that *structure*. A *relation* is defined as a fixed set of *data attributes*. Every *tuple* is an ***instance*** of a specific *relation* and shares the same static *structure* defined by that *relation* with every other *tuple* of that *relation*. The *relation* concept thereby fuses the existence of the *tuples* to that of their *relation*; *tuples* cannot exist independent of their defining *relation*. *Tuples* are said to be ***members of*** their *relation*. *Tuples* are added to or deleted from their *relation*. The order of attributes in a *relation* is insignificant except that the order is consistent for all *tuples*. *A relation* is also commonly called a ***table*** and each of its *instances*, a ***row***. The collection of every *data attribute value(s)* for a particular *data attribute* in a *table* is called a ***column***.

### 4. Relationships

Relationships in the *relational* paradigm are based on the property of *remembrance* and the juxtaposition of *data attribute values* in one or more tuples in the same or across *relations*.



#### 4.1. Behavioral Relationships

The behavioral relationships are all based upon the *data attribute value(s)* and which values are permitted to coexist in and across *tuples* and *relations*.

##### 4.1.1. Functional Dependency

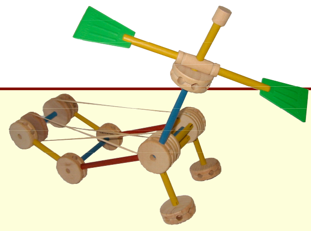
In a *relation* a *data attribute* is ***functionally dependent*** when its *data attribute value* is always the same in any *tuple* for a given value in a second *data attribute*. In other words, the value of the first *data attribute* ***is determined by*** the value of the second (called the ***determinant***). *Functional dependency* expresses the informational integrity of *relations*.

###### 4.1.1.1. Entity Integrity

***Entity integrity*** defines the two-fold quality of *tuple* uniqueness in a *relation*: a) every *tuple* in a *relation* is distinct in some *data attribute* *value(s)* from every other *tuple* in that *relation* or symmetrically, b) there is a designated subset of *data attributes* (*column(s)*) called the ***primary key*** such that the *data attribute value(s)* in that *relation* is distinct for all *tuples* and no values may be ***null*** (a value which is unknown and incomparable to any other value). There may be more than one subset of *data attributes* with the value characteristics of the *primary key* (each called a ***candidate key***) but only one is designated as the *primary key*.

#### 4.1.2. Association

An ***association*** is a relationship between *tuples* in the same or different *relations*. *Tuples* are intrinsically separable by way of *entity integrity*. At the same time, humans are compelled to categorize their experience of things in the physical world by superimposing groupings that collect *tuples* into sets. *Tuples* become members in a group based upon *data attribute value(s)*. This property is called ***membership IN***. This property also permits humans to identify a *tuple* that is not in a set (i.e. discrimination). (*Membership IN an association is distinct from membership OF a relation which is intrinsic by way of instance relationship.*)



**Without syntax?**

Every *language* that is invented to express concepts carries with it the understanding and the biases of the inventor. Depending on his/her purpose(s) those biases simplify certain tasks performed with the language but may obscure the underlying concepts.

Programming language design must deal with the feasibility of automated translation and interoperability with other programming languages and operating systems. Compromises and assumptions are chosen to make the resulting language efficient, effective and marketable.

The goal of this description of the entity-relationship paradigm is to succinctly make the concepts understandable - an ambitious task to say the least!

*- Professor Waguespack*

##### 4.1.2.1. Relational Operations

Membership IN is realized through **relational operations** keying on relation structure and values. Each relational operation produces a real or virtual relation as its result. The **selection** operation retrieves *tuple*(s) based upon a **selection predicate** testing data attribute value(s) to determine whether each *tuple* is or is not in the set. Selection predicates are based on any boolean comparison including constant values or values referenced in *data attribute value(s)*. The **projection** operation copies all the *data attribute value(s)* for a particular *column(s)*.

*Association* between *relations* (or a *relation* and itself) is based upon relating (matching) *data attribute values* in *tuples* of one *relation* with those of another. The ***join*** operation pairs every combination of *tuples* from one *relation* with those of another *relation* and copies the *data attribute values* from the pairs where the pairing satisfies a *selection predicate*. This *relational operation* is called *join* because facts from two sources are joined in the result.

##### 4.1.2.2. Join Compatibility

***Join compatibility*** requires that the values involved in comparisons (i.e. *selection predicates*) whether constants or *data attribute values* derive from the same *data attribute domain*.

##### 4.1.2.3. Referential Integrity

When *relations* are devised such that a *tuple* in one *relation* predisposes the existence of (***owns***) *tuple(s)* in another, the *data attribute(s)* of the second required to *join* the *relations* is called a ***foreign key***. Referential integrity asserts that any value found in the *data value attribute(s)* of a *foreign key* must appear in a *tuple* of the first *relation* as the value of a *candidate key* or itself be *null*.

#### 4.1.3. Normalization

*Relational* model consistency depends on the semantic concurrence of the behavioral relationships and the objectives of the database modeler, the ***intension,*** (rather than the accident of a *relation's* contents at any particular instant, its ***extension***). The integrity properties defined above enable the database modeler to devise a structure and behavior of *relations* that avoid semantic discord called ***anomalies***, the unintended loss or modification of information by relational operations. *Relations* designed to avoid certain kinds of *anomalies* are said to be ***normalized*** or in ***normal form***. ***Normalization*** is the arrangement of *data attributes* and their relationships among *relation* structures to prevent particular *anomalies*.

##### 4.1.3.1. First Normal Form

First Normal Form asserts that every *data attribute value* is ***atomic***, indivisible in value or form and may not be operated upon except as a whole and single value.

##### 4.1.3.2. Second Normal Form

Second Normal Form is first normal form and asserts that every *data attribute value* not in the primary key is ***fully functionally dependent*** upon the *primary key*. ("Fully" means applying to every *data attribute* of the *primary key*.)

##### 4.1.3.3. Third Normal Form

Third Normal Form presupposes first and second normal forms and asserts that no *data attribute* outside the *primary key* is ***transitively dependent*** upon the *primary key*. ("Transitively" means an attribute(s) *functionally dependent* upon an attribute *functionally dependent* upon an attribute (. . .) *functionally dependent* on the *primary key*.)