Codd’s Rules & Noramalization

Overview

- Codd's twelve rules are a set of thirteen rules (numbered zero to twelve) proposed by Edgar F Codd, a pioneer of the relational model for databases.
- It is designed to define what is required from a Database Management System in order for it to be considered *relational*, i.e., a Relational Database Management System (RDBMS).
- They are sometimes jokingly referred to as "Codd's Twelve Commandments".

Rules…

- **Rule 0: The *Foundation rule*:**

  For any system that is advertised as, or claimed to be, a relational data base management system, that system must be able to manage data bases entirely through its relational capabilities.

Rules…

- Rule 1: *The information rule:*

  All information in a relational data base is represented explicitly at the logical level and in exactly one way — by values in tables.

- Rule 2: *The guaranteed access rule:*

  Each and every data (atomic value) in a relational data base is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name.

Rules…

- Rule 3: *Systematic treatment of null values:*

  Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type.

Rule 4: *Dynamic online catalog based on the relational model:*

The data base description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data.

Rules…

- **Rule 5: The *comprehensive data sublanguage rule*:**

  A relational system may support several languages and various modes of terminal use (for example, the fill-in-the-blanks mode). However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings and that is comprehensive in supporting all of the following items:

- **Data definition.**
- **View definition.**
- **Data manipulation (interactive and by program).**
- **Integrity constraints.**
- **Authorization.**
- **Transaction boundaries (begin, commit and rollback).**

Rules…

- **Rule 6: The view *updating rule*:**

  All views that are theoretically updatable are also updatable by the system.

- **Rule 7: *High-level insert, update, and delete*:**

  The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data.

Rules…

- Rule 8: Physical data independence:
- Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representations or access methods.
- Rule 9: Logical data independence:
- Application programs and terminal activities remain logically unimpaired when information-preserving changes of any kind that theoretically permit un-impairment are made to the base tables.
- Rule 10: Integrity independence:
- Integrity constraints specific to a particular relational data base must be definable in the relational data sub-language and storable in the catalog, not in the application programs.

Rules…

- **Rule 11: *Distribution independence*:**

  A relational DBMS has distribution independence.

- **Rule 12: The *non-subversion rule*:**

  If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher level relational language (multiple-records-at-a-time).

Normalization

If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator. Managing a database with anomalies is next to impossible.

- Update anomalies – If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.
- Deletion anomalies – We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.
- Insert anomalies – We tried to insert data in a record that does not exist at all.

First Normal Form

- **First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.**

FNF (cont.)

We re-arrange the relation (table) as below, to convert it to First Normal Form

Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

Each attribute must contain only a single value from its pre-defined domain

Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

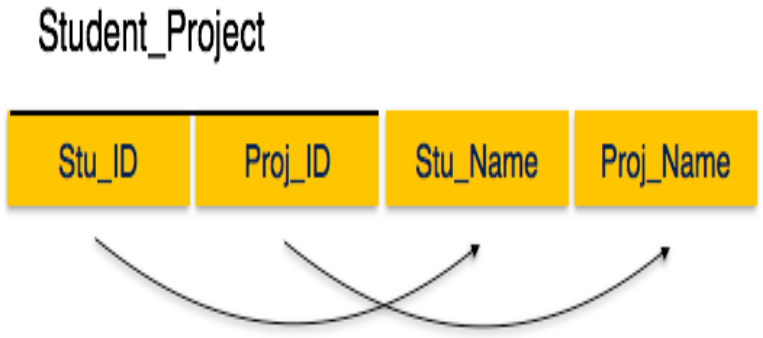
Second Normal Form

Before we learn about the second normal form, we need to understand the following –

- Prime attribute – An attribute, which is a part of the prime-key, is known as a prime attribute.
- Non-prime attribute – An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

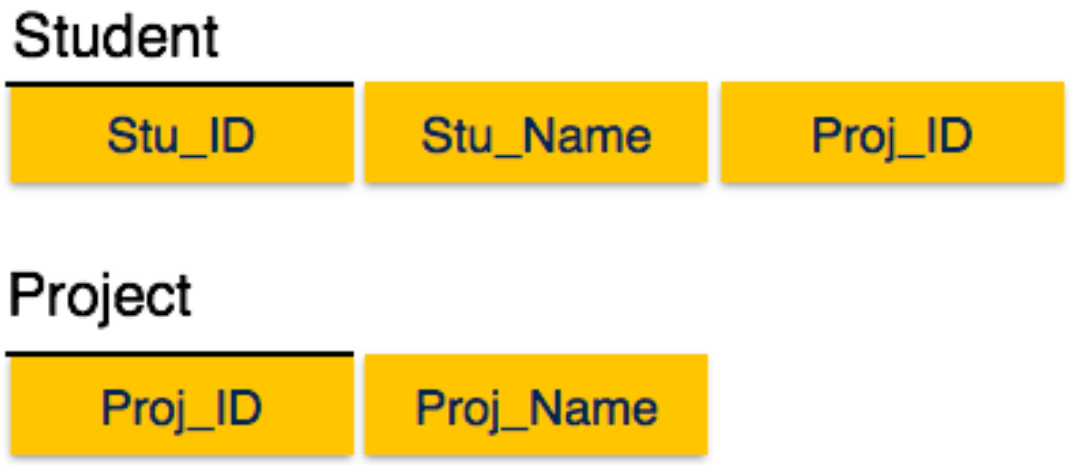
If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if $X \rightarrow A$ holds, then there should not be any proper subset Y of X, for which $Y \rightarrow A$ also holds true.

SNF (cont.)



- We see here in Student_Project relation that the prime key attributes are Stu_ID and Proj_ID. According to the rule, non-key attributes, i.e. Stu_Name and Proj_Name must be dependent upon both and not on any of the prime key attribute individually. But we find that Stu_Name can be identified by Stu_ID and Proj_Name can be identified by Proj_ID independently. This is calledpartial dependency, which is not allowed in Second Normal Form.

SNF (cont.)



- We broke the relation in two as depicted in the above picture. So there exists no partial dependency.

Third Normal Form

- **For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy –**
- **No non-prime attribute is transitively dependent on prime key attribute.**
- **For any non-trivial functional dependency, X → A, then either –**
  - X is a superkey or,
  - A is prime attribute.

TNF (cont.)

We find that in the above Student_detail relation, Stu_ID is the key and only prime key attribute. We find that City can be identified by Stu_ID as well as Zip itself. Neither Zip is a superkey nor is City a prime attribute. Additionally, Stu_ID → Zip → City, so there exists **transitive dependency**.

To bring this relation into third normal form, we break the relation into two relations as follows −