Evaluación de la calidad del diseño

CI-0127 Bases de Datos, Universidad de Costa Rica

Sivana Hamer

Importante: Este documento recopila contenidos de diversos de sitios web especializados, académicos y documentos compartidos por universidades. Toda la información es utilizada con fines estrictamente académicos. Para más información, puede ver las referencias.

1 Design guidelines

While designing a database, we have yet to formally define the goodness or appropriateness of the design. Thus, we can use both guidelines and measure the quality of the design. There are two levels we can define goodness:

1. The *logical or conceptual level* on how users interpret the schemas.
2. The *implementation or physical level* on how tuples are stored and updated.

The goals are:

- **Information preservation:** Preserves the information (e.g., attributes, entities, relationships) described in the models from the conceptual design to the logical design mapping.
- **Minimum redundancy:** Minimizes saving the same information to reduce storage, inconsistencies and the need of multiple updates. Though, sometimes some form of redundancy may be desired depending on the application (NoSQL).

Following, we shall define good general informal guidelines for design.

Guideline #1: Clear semantics

We can informally measure how *semantically* easy to understand are the database elements (entities and relationship) in the design. Thus, every element should have a straightforward meaning that represents only one concept.

For example, the following table combines multiple concepts of students, schools and faculty into only one relation named *STUDENT_SCHOOL_FACULTY*. Therefore, it is hard to understand which attributes are related to which concept. For example, is the student's name, the school's name or the faculty's name? Is the number of students for the school or the faculty? Which acronym are they referring to?

STUDENT_SCHOOL_FACULTY

Email	Name	Phone_number	Acronym	Number_students

Guideline #2: Avoid redundancy

If data in the database is redundant (storing the result of natural joins), we are wasting space and may lead to several anomalies. For example, the following table shows how the name, phone and

faculty name for the "ECCI"" school is duplicated in t_1 and t_2. Furthermore, the name of "Bob Benavidez" is repeated, though they can be recognized only with their email.

STUDENT_SCHOOL_FACULTY

Student_mail	Student_name	School_acronym	School_phone	Faculty_name
alicia.armando23	Alicia Armando	ECCI	2511-8000	Ingenería
bob.benavidez	Bob Benavidez	ECCI	2511-8000	Ingenería
bob.benavidez	Bob Benavidez	EMat	2511-6551	Ciencias Básicas
carlos.calvo	Carlos Calvo	EAN	2511-9180	Ciencias Sociales
daniela.delgado	Daniela Delgado	EMat	2511-6551	Ciencias Básicas

There are different types of anomalies that can occur:

- **Insertion:** While inserting tuples, inconsistencies can occur with new data that is added. For example, when we add another student of the "ECCI"" school, we must ensure that the phone number and faculty name are correct in all the tuples. Furthermore, if we want to add information only about the school without students it is not possible due to key constraints.
- **Deletion:** While deleting certain tuples, we can lose all the information in the database. For example if we delete "Carlos Calvo", we lose all the information related to the school "EAN".
- **Modification:** While modifying tuples, we have to always modify all redundant references. For example, if we modify the phone number of the "ECCI"" school we must update the phone number for tuple t_1 and t_2 else data will be inconsistent.

Guideline #3: Reduce NULLs

With many tuples not applying in a relation, space can be wasted and the differences between NULL types may not be understandable (does not apply, unknown or absent). For example, in the following table saving in an attribute the position of a student in the association will probably lead to many NULLs as there are less than ten members per year. Furthermore, Fax information probably does not exist for most students. Their grade in kindergarten is probably not known for most students. Finally, most students will also probably not link their instagram account within the institution.

STUDENT

Email	Name	Position_association	Fax_number	Kindergarden_grade	Instagram

Guideline #4: Consider false tuples

We must ensure when tables are created, the result of joining them gives a correct result. Thus, we must use attributes that are appropriately related such as keys and foreign keys. If not, it generates false tuples that do not represent valid data (a.k.a sussy). For example, if we have the following tables *STUDENT* and *SCHOOL*. The "correct" tuples are shown for guideline #2.

STUDENT

Email	Name	Faculty_name
alicia.armando23	Alicia Armando	Ingenería
bob.benavidez	Bob Benavidez	Ingenería
bob.benavidez	Bob Benavidez	Ciencias Básicas
carlos.calvo	Carlos Calvo	Ciencias Sociales
daniela.delgado	Daniela Delgado	Ciencias Básicas

If we join both tables on the attribute *Faculty_name*, we will generate a false tuple marked in red that did not previously exist.

SCHOOL

School_acronym	School_phone	Faculty_name
ECCI	2511-8000	Ingenería
EMat	2511-6551	Ciencias Básicas
EAN	2511-9180	Ciencias Sociales
ECCC	2511-3600	Ciencias Sociales

STUDENT_SCHOOL_FACULTY

Student_mail	Student_name	School_acronym	School_phone	Faculty_name
alicia.armando23	Alicia Armando	ECCI	2511-8000	Ingenería
bob.benavidez	Bob Benavidez	ECCI	2511-8000	Ingenería
bob.benavidez	Bob Benavidez	EMat	2511-6551	Ciencias Básicas
carlos.calvo	Carlos Calvo	EAN	2511-9180	Ciencias Sociales
carlos.calvo	Carlos Calvo	ECCC	2511-3600	Ciencias Sociales
daniela.delgado	Daniela Delgado	EMat	2511-6551	Ciencias Básicas

All guidelines

Guideline #1: While designing the relational schema, the attributes should be easy to understand and explain. Thus, every entity or relationship has a straightforward meaning representing only one concept.

Guideline #2: Design to avoid insertion, deletion or modification anomalies. If not possible, clearly state the anomalies and update the database correctly.

Guideline #3: Avoid NULL attributes while possible. If not possible, ensure that they do not apply to the majority of tuples in the relation.

Guideline #4: Design relation schemas so that they can be joined correctly, using the appropriate primary and foreign keys that do not generate false tuples.

2 Functional dependencies

Based on these informal rules, we can use functional dependencies to formally define some of these issues.

A *functional dependency* is a constraint denoted by $X \rightarrow Y$ for a set of attributes X and Y for a relation R.It can be defined when any two tuples that have $t_1[X] = t_2[X]$ must also have $t_1[Y] = t_2[Y]$.

- The attributes of Y are determined (i.e., functionally dependent) on the set of attributes X.
- It can be abbreviated into FD or f.d.
- $X \rightarrow Y$ does not indicate that $Y \rightarrow X$ is true.
- All candidate keys X can determine the attributes of any attribute Y in R. Therefore, $X \rightarrow R$.
- This constraint is for any possible relation state r of R. Thus, it is a property of the relation R and not for a state.
- We *cannot* infer automatically a FD based on the state of a relation, but we may indicate that a DF *may* exist. Any counter example in a state (old, current or old) can disprove the DF.

3 Normal forms

A schema can be normalized based on a series of tests that specify criteria for a *normal form*.

- Combining normal forms, ensuring to not create false tuples (*nonadditive join or lossless join*) and preserving most dependencies (*dependency preservation property*), we can improve the database design. We do not need to ensure the dependency preservation property in the resulting design.
- The process in which we pass a non-compliant design to a normal form is called *normalization*. The inverse process, in which a design normal form is lowered is called *denormalization*.
- There are various normal forms, but we shall see in detail the first three: 1NF, 2NF and 3NF.
- Though some normal forms do not need to satisfy the requirements of a lower level normal form, for historical reasons it is customary that the sequence is followed. Thus, 3NF already satisfies 1NF.

1NF

Every attribute within a relation must be *atomic* (a single value for any attribute) or nested relations (tuples with relations within it). This is considered a requirement for any flat relational model. Every attribute that does not follow this restriction is moved to another relation that has a foreign key to the primary key of this relation. Thus separating the non-1NF relation into two 1NF relations.

2NF

Every attribute that is part of a candidate key is a *prime attribute*. While, attributes are non prime if they are not prime. 2NF expects every non prime attribute in a relation to only be fully dependent on the primary key of the relation. Thus, for every primary key a new relation is created with all the nonprime attributes that depend on that primary key (based on the FD).

3NF

There is a transitive dependency from $X \to Z$, when there exists a FD from $X \to Y$ and $Y \to Z$ where Y is a non prime attribute.

3nd expects that every non prime attribute in a relation has a FD with the primary key. As in, there can be no transitive dependencies from the primary key of the relation. To normalize this dependency, we create a new relation with the primary key as the Y in the transitive relation and leave the attribute in the previous relation.

References

[1] R. Elmasri and S. Navathe, *Fundamentals of database systems*, 7th ed. Pearson, 2016, chapters 14 and 15.

[2] C. Faloutsos. Schema refinement and normalization. [Online]. Available: https://15415.courses.cs.cmu.edu/fall2016/syllabus.html

