Normalization

- A process to design a relational schema based on relational theory
- Provides guidelines for relational database design that…
    - Minimize redundancy
    - Avoid potential inconsistency
    - Avoid anomalies
        - Insertion anomalies
        - Deletion anomalies
        - Update anomalies
    - 1NF, 2NF, 3NF…
        - Higher normal forms are more restrictive
        - If a table is in a higher NF, then it is also in all of the lower NFs

Before a table can be in a higher normal form, it must first satisfy the criteria for the lower normal forms.

Normal Forms: First Normal Form (1NF)

- A relation R is in 1NF, if all attributes have atomic values
    - Atomic values – one value for an attribute
        - No repeating groups
        - No multi-valued attributes
        - No composite attributes

If a table has any repeating groups and/or multi-valued attributes and/or composite attributes, then it violates first normal form (1NF).

Example

emp(eid, ename, address, skill1, skill2, skill3, skill4, degree)

In the above example, the table emp is in violation of 1NF because it has repeating groups, composite attributes and a multivalued attribute.

To get it into 1NF, we must address each of the issues that make emp violate 1NF. We start by addressing the issue of repeating groups.

HINT: You can typically identify repeating groups because they tend to be listed with the same field name but with sequential numbers at the end of the field name (to distinguish one from the other). In the above example, you can readily identify the repeating group skill. Look for that sort of pattern to help you identify repeating groups.

The repeating group would have to be removed from the original table and become a field in a separate table. There will be a 1:N relationship between the original table and the new table. The new table will have a composite primary key (PK) that consist of the PK of the original table plus the attribute that was repeating in the original table. The primary key of the original table will also be a foreign key (FK) in the new table.

In the original table, the repeating group would have had multiple columns for each value that the repeating group could have, but in the new table each value will be represented by a row of data.

This is good from a design point of view because in the original table, if a person had a new skill that wasn't already accounted for in the first four columns for skill, then the table would have had to be modified to add an additional column. Clearly, that is not a desirable approach. However, in the new table, if an employee has a new skill, then it is merely a matter of inserting a new row in the new table….no design changes to the table would be required to accommodate recording the new skill.

ORIGINAL TABLE:

emp(eid, ename, address, skill1, skill2, skill3, skill4, degree)

INTERMEDIATE RESULTS:
emp(eid, ename, address, degree)

empSkill(eid, skill)

Note that the above represents an intermediate result that only addresses the repeating groups. There are still issues with emp in the INTERMEDIATE RESULT. Specifically, ename and address are composite attributes AND degree is a multivalued attribute.

Emp(eid, ename, address, degree)

In the above example, degree is a multi-valued attribute because each employee can hold multiple degrees at any given time (e.g. Associates, Bachelors, Master's, Ph.D.). So, the solution is to remove the multi-valued attribute from the table and put it in a separate table with the PK of the original table as a FK. Additionally, the new table will have a composite PK that consists of the PK of the original table and the attribute that was a multi-valued attribute in the original table. There will be a 1:N relationship between the original table and the new table.

INTERMEDIATE RESULTS:

emp(eid, ename, address)

empDegree(eid, degree)

empSkill(eid, skill)

The above are the intermediate results of getting the original table into 1NF. There is, however, one more issue with the emp table…composite attributes. The next slide addresses composite attributes.

emp(eid, ename, address)

In the above, ename and address are composite attributes. The solution will result in additional fields in the original table (no new tables will be created to address composite attributes). So, the attribute for employee name, ename, will be decomposed into an attribute for first name (fname) and last name (lname). Similarly, address will be decomposed into its parts: street, city, state and zip.

So, the final solution for the original table:

emp(eid, ename, address, skill1, skill2, skill3, skill4, degree)

Is:

emp(eid, fname, lname, street, city, state, zip)

empSkill(eid, skill)

empDegree(eid, degree)

Normal Forms: Second Normal Form (2NF)

- The relation Database is in (2NF) when it is in 1NF and all non-key attributes are fully functionally dependent on the entire primary key
- Use FDs to reach 2NF
    - Each determinant becomes the primary key in a relation
    - All the attributes that are functionally dependent upon the determinant become non-key attributes in the relation

Example

- worker_proj(wid, pnumber, hours, lname, pname)
- Worker_proj(wid, pnumber, hours)
- worker(wid, lname)
- project(pnumber, pname)

In the example above, worker_proj contains a composite primary key that consists of wid (worker ID) and pnumber (project number). The only attribute that depends on the entire composite primary key (wid, pnumber) is hours. So, hours would remain in the original table along with the composite primary key. Looking at the other nonkey attributes (i.e. the descriptive attributes), however, it is clear that the attribute for last name (lname) does NOT depend upon the entire primary key. In other words, the employee's last name does NOT depend upon the project number attribute (pnumber). Rather, the employee's last name only depends upon the worker ID (wid). Similarly, the project name attribute (pname) does NOT depend upon the worker ID attribute (wid), but rather on the project number (pnumber). Therefore, worker_proj is in violation of the second normal form (2NF) because it contains attributes that only depend on part of the primary key instead of depending on the *entire* primary key.

Therefore, the attributes that do not depend on the entire attribute should be removed from the original table and placed in a separate table with the PK that it depends on. For example, lname would be removed from the original table and placed into a separate table (worker) with the attribute that it depends on (wid). WID would be the primary key of the new table. Similarly, pname would be removed from the original table and placed in a separate table (project) with the primary key that it depends on (pnumber).

Just to restate the original problem in terms of functional dependency:

wid, pnumber → hours, lname, pname FALSE

We used the determinant (wid, pnumber) to identify which attributes it determines because those attributes will remain the table.

wid, pnumber → hours TRUE

wid, pnumber → lname FALSE

wid, pnumber → pname FALSE

For those attributes that are not determined by the determinant (i.e. the left-hand side of the equation), we removed the attributes from the original table that caused the functional dependencies to be false. We created separate tables where the functional dependencies were true.

wid → lname TRUE

pnumber → pname TRUE

The Database is in Third Normal Form(3NF) when:

- The relation is in 2NF and there are no transitive dependencies
- Transitive dependency:
    - A → B and B → C therefore A → C
- Easier definition to understand
    - A relation is in 3NF, if every attribute is directly dependent on its key attribute