Database redesign is necessary:

- To fix the mistakes made during the making of database design
- To change in system requirements

Correlated subquery: it can be used to get the functional dependency.

SQL exists and not exists:

Ex. You can use it too for takin the functional dependency.

Before making a redesign to a database you must:

- Understand the structures and the contents of the database before making any structure changes.
- Test the new changes on a test database before making real changes.
- Create a backup of the operational database before making any structure changes.

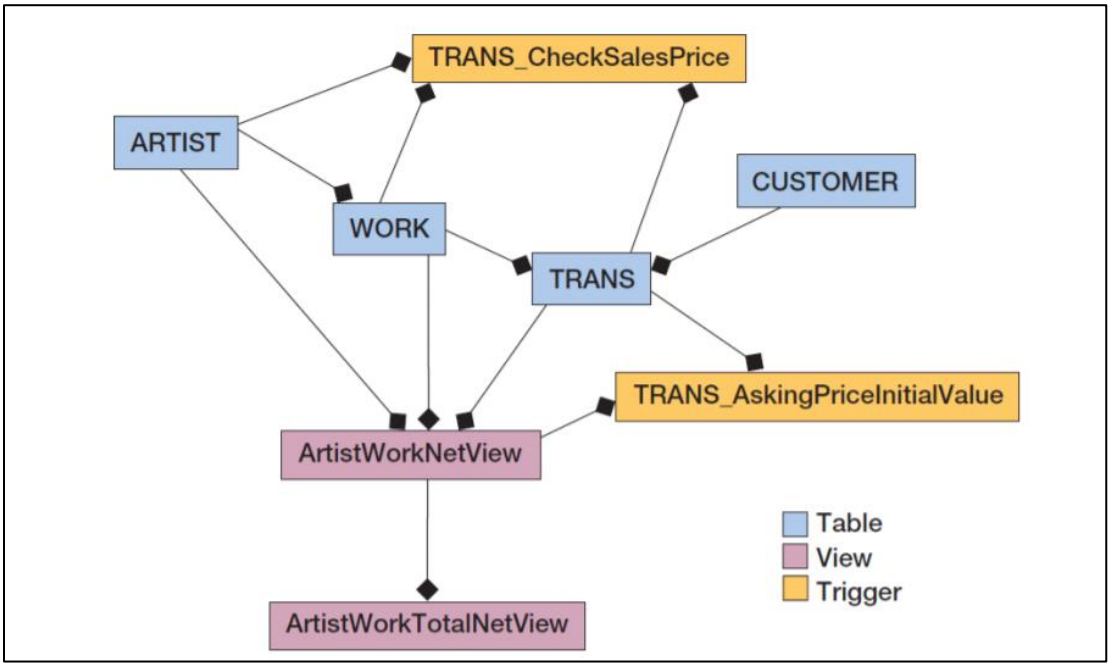
Reverse engineering:

It's the process which make us create the data model from a database to begin the redesign project.

Dependency graphs:

They are the diagrams that are used to show the dependency of one element on another.

Ex.



Testing a database:

To test a database for a redesign process you must do three different copies:

- A small test database for initial testing.
- A large test database for secondary testing.
- The operational database.

Database redesign changes

We use redesign process to change:

1- tables and columns

- Changing table names.
- Adding or dropping table columns.
- Changing datatype or constraints.
- Adding or dropping constraints.

2- relationships

- Changing cardinalities.
- Adding and deleting relationships.
- Adding and removing relationships for denormalization.

NOTES: if the source table name was changed, you must change the view table.

When changing a datatype ex. Int, date, etc. to varchar or char will usually succeed but when changing from varchar or char to int, date, etc. can be a problem.

Changing minimum cardinalities:

To change from the parent side:

- Changing from O to M, change the foreign key constraint from NULL to NOT NULL.
- Changing from M to O, change the foreign key constraint from NOT NULL to NULL.

To change from the child side:

- Changing from O to M, add a trigger that enforce the constraint.
- Changing from M to O, drop the trigger that enforce the constraint.

How to change maximum cardinalities:

Changing maximum cardinality 1:1 to 1:N:

- If the foreign key is in the correct table, remove the unique constraint on the foreign key column.
- If the foreign key is in the wrong table, move the foreign key to the correct table and do not place a unique constraint on that table.

Changing maximum cardinality 1:N to N:M:

- Build a new intersection table and move the key and foreign key values to the intersection table.

When you reduce the cardinalities, it may result in data loss.

Reducing N:M to 1:N:

- Create a foreign key in the parent table and move one value from the intersection table into that foreign key.

Reducing 1:N to 1:1:

- Remove any duplicates in the foreign key and then set a uniqueness constraint on that key.

Adding and deleting relationships:

Adding new tables and relationships:

- Add the tables and relationships using CREATE TABLE statements with FOREIGN KEY constraints.
- If an existing table has a child relationship to the new table, add a FOREIGN KEY constraint using the existing table.

Deleting relationships and tables:

- – Drop the foreign key constraints and then drop the tables.

Forward engineering: the process when you apply the changes data model changes to an existing database.

NOTE: the forward engineering should be tested before applying it to the operational database.