IX. Normalization

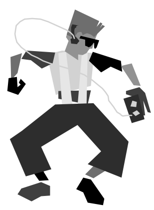
Database Redundancies and Anomalies
Functional Dependencies
The Boyce-Codd Normal Form
Lossless Decompositions
Preserving Dependencies
The Third Normal Form
Normal Forms and Database Design

Seminar on Databases

Martedi, 16 Ottobre, ore 11.30
Aula Seminari di Matematica

Klaus R. Dittrich, University of Zurich, Switzerland

SINGAPORE:
Towards flexible querying of heterogeneous data sources



Normal Forms and Normalization

- A ***normal form*** is a property of a database schema.
- When a database schema is un-normalized (that is, does not satisfy a normal form), it allows redundancies of various types which can lead to anomalies and inconsistencies.
- Normal forms can serve as basis in order to carry out quality analysis for a given database schema and constitutes a useful tool for database design.
- ***Normalization*** is a procedure that transforms an un-normalized schema into a normalized one.

Example of Redundancies

Employee	Salary	Project	Budget	Function
Brown	20	Mars	2	technician
Green	35	Jupiter	15	designer
Green	35	Venus	15	designer
Hoskins	55	Venus	15	manager
Hoskins	55	Jupiter	15	consultant
Hoskins	55	Mars	2	consultant
Moore	48	Mars	2	manager
Moore	48	Venus	15	designer
Kemp	48	Venus	15	designer
Kemp	48	Jupiter	15	manager

Anomalies

The value of the salary of an employee is repeated in every tuple where the employee is mentioned, leading to a ***redundancy***. Redundancies lead to anomalies:

- If the salary of an employee changes, we have to modify the value in all the corresponding tuples (***update anomaly***.)
- If an employee ceases to work in projects but does not leave the company, all the corresponding tuples are deleted, leading to loss of information (***deletion anomaly***.)
- A new employee cannot be inserted in the relation it until the employee is assigned to a project (***insertion anomaly***.)

Normalization -- 5

What's Wrong???

- We are using a single relation to represent data of very different types.
- In particular, we are using a single relation to store the following types of entities, relationships and attributes:
  - ✓ Employees and their salaries;
  - ✓ Projects and their budgets;
  - ✓ Participation of employees in projects, along with their functions.
- To set the problem on a formal footing, we introduce the notion of ***functional dependency***.

Normalization -- 6

Functional Dependencies in the Example

- Each employee has a unique salary. We represent this dependency as

  Employee $\rightarrow$ Salary

  and we say that Salary ***functionally depends*** Employee.
- This means that there exists a function that takes an Employee atrribute value and returns a (single) value for the attribute Salary.
- Likewise,

  Project $\rightarrow$ Budget

  i.e., each project has a unique budget; or, Budget functionally depends on Project.

Functional Dependencies

- Given a schema $R(X)$ and two non-empty subsets Y and Z of the attributes X, we say that there is a ***functional dependency*** between Y and Z $(Y \rightarrow Z)$, if for every relation instance r of $R(X)$ and every pair of tuples t_1 and t_2 of r such that $t_1.Y = t_2.Y$, it is the case that $t_1.Z = t_2.Z$.
- A functional dependency is a statement about ***all allowable relations*** for a given schema.
- Functional dependencies have to be identified by understanding the semantics of the application.
- Given a particular relation r_0 of $R(X)$, we can tell if a dependency holds or not; but just because it holds for r_0, doesn't mean that it also holds for $R(X)$!

Non-Trivial Dependencies

- A functional dependency $Y \rightarrow Z$ is ***non-trivial*** if no attribute in Z appears among the attributes of Y, e.g.,
  - ✓ Employee → Salary is non-trivial;
  - ✓ Employee Project →Project is trivial.
- In our example, anomalies arise precisely for the attributes which are involved in functional dependencies; specifically
  - ✓ Employee → Salary;
  - ✓ Project → Budget.
- Moreover, note that the example includes another functional dependency:
  - ✓ Employee, Project → Function.

Dependencies Cause Anomalies, ...Sometimes!

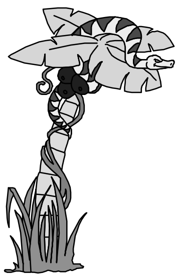
- The first two dependencies cause undesirable redundancies and anomalies.
- The third dependency, however, does not cause redundancies because {Employee, Project} constitute a key of the relation (...and a relation cannot contain two tuples with the same values of these attributes.)

Dependencies on keys are OK,

other dependencies are not!

Boyce–Codd Normal Form (BCNF)

- A relation r is in ***Boyce–Codd Normal Form*** if for every (non-trivial) functional dependency $X \rightarrow Y$ defined on it, X contains a key K of r. That is, X is a superkey for r.
- Anomalies and redundancies, as discussed above, do not occur in databases with relations in Boyce–Codd normal form.



Normalization Through Decomposition

- A relation that is not in Boyce–Codd normal form, can be replaced with one or more normalized relations using a process called ***normalization***.
- We can eliminate redundancies and anomalies for the example relation if we replace it with the three relations, obtained by projections on the sets of attributes corresponding to the three functional dependencies.
- The keys of the relations we obtain are the left hand side of a functional dependency: the satisfaction of the Boyce–Codd normal form is therefore guaranteed for the new relations..

Example of Normalization

Employee	Salary
Brown	20
Green	35
Hoskins	55
Moore	48
Kemp	48

Project	Budget
Mars	2
Jupiter	15
Venus	15

Employee	Project	Function
Brown	Mars	technician
Green	Jupiter	designer
Green	Venus	designer
Hoskins	Venus	manager
Hoskins	Jupiter	consultant
Hoskins	Mars	consultant
Moore	Mars	manager
Moore	Venus	designer
Kemp	Venus	designer
Kemp	Jupiter	manager

Another Example

Employee	Project	Branch
Brown	Mars	Chicago
Green	Jupiter	Birmingham
Green	Venus	Birmingham
Hoskins	Saturn	Birmingham
Hoskins	Venus	Birmingham

This relation satisfies the functional dependencies:

- ✓ Employee → Branch
- ✓ Project → Branch

A Possible Decomposition

Employee	Branch
Brown	Chicago
Green	Birmingham
Hoskins	Birmingham

Project	Branch
Mars	Chicago
Jupiter	Birmingham
Saturn	Birmingham
Venus	Birmingham

The Join of the Projections

Employee	Project	Branch
Brown	Mars	Chicago
Green	Jupiter	Birmingham
Green	Venus	Birmingham
Hoskins	Saturn	Birmingham
Hoskins	Venus	Birmingham
Green	Saturn	Birmingham
Hoskins	Jupiter	Birmingham

The result of the join is different from the original relation.

We lost some information during the decomposition!

Lossless Decomposition

- The decomposition of a relation *r* on X_1 and X_2 is ***lossless*** if the join of the projections of *r* on X_1 and X_2 is equal *to r* itself (that is, does not contain ***spurious*** tuples).
- It is clearly desirable to allow only lossless decompositions during normalization.

A Condition for Lossless Decomposition

- Let *r* be a relation on *X* and let X_1 and X_2 be two subsets of X such that $X_1 \cup X_2 = X$. Furthermore, let $X_0 = X_1 \Upsilon\ X_2$.
- If r satisfies the functional dependency $X_0 \rightarrow X_1$ or the functional dependency $X_0 \rightarrow X_2$, then the decomposition of r on X_1 and X_2 is lossless.
- In other words, *r* has a lossless decomposition on two relations if the set of attributes common to the relations is a key for at least one of the decomposed relations.

A Lossless Decomposition

Employee	Branch
Brown	Chicago
Green	Birmingham
Hoskins	Birmingham

Employee	Project
Brown	Mars
Green	Jupiter
Green	Venus
Hoskins	Saturn
Hoskins	Venus

Another Problem...

- Assume we wish to insert a new tuple that specifies that employee Armstrong works in Birmingham and participates in project Mars.
- In the original relation an update of this kind would be immediately identified as illegal, because it would cause a violation of the Project $\rightarrow$ Branch dependency.
- For the decomposed relations, however, this is not possible because the two attributes Project and Branch have been moved in different relations.

Preserving Dependencies

- A decomposition ***preserves dependencies*** if each of the functional dependencies of the original relation schema involves attributes that appear together in one of the decomposed relation schemas.
- It is clearly desirable that a decomposition preserves dependencies because then it is possible to ensure that the decomposed schema satisfies the same constraints as the original schema.

Desirable Qualities for Decompositions

- Decompositions should always satisfy the properties of lossless decomposition and dependency preservation:
  - ✓ ***Lossless decomposition*** ensures that the information in the original relation can be accurately reconstructed based on the information represented in the decomposed relations.
  - ✓ ***Dependency preservation*** ensures that the decomposed relations have the same capacity to represent the integrity constraints as the original relations and therefore to reveal illegal updates.

A Relation not in BCNF

Manager	Project	Branch
Brown	Mars	Chicago
Green	Jupiter	Birmingham
Green	Mars	Birmingham
Hoskins	Saturn	Birmingham
Hoskins	Venus	Birmingham

Assume the following dependencies:

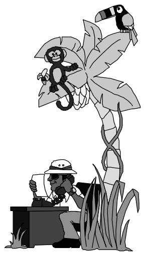
- Manager $\rightarrow$ Branch -- each manager works in a particular branch;
- Project, Branch $\rightarrow$ Manager -- each project has several managers, and runs on several branches; however, a project has a unique manager for each branch.

A Problematic Decomposition

- The relation is not in Boyce–Codd normal form because the left hand side of the first dependency is not a superkey.
- At the same time, no good decomposition of this relation is possible: the dependency Project Branch $\rightarrow$ Manager involves all the attributes and thus no decomposition is able to preserve it.
- We conclude that sometimes Boyce–Codd normal form cannot be achieved for a particular relation and set of functional dependencies without violating the principles of lossless decomposition and dependency preservation.

A New Normal Form

- A relation *r* is in ***third normal form*** (***3NF***) if, for each (non-trivial) functional dependency $X \rightarrow Y$, at least one of the following is true:
  - ✓ *X* contains a key *K* of *r*;
  - ✓ Each attribute in *Y* is contained in at least one key of *r*.



BCNF and 3NF

- The previous schema is not in Boyce–Codd normal form, but it ***is*** in third normal form.
- In particular, the Project, Branch $\rightarrow$ Manager dependency has as its left hand side a key, while Manager $\rightarrow$ Branch has a unique attribute for the right hand side, which is part of the {Project, Branch} key.
- The third normal form is less restrictive than the Boyce–Codd normal form and for this reason does not offer the same guarantees of quality for a relation; it has the advantage however, of always being achievable.

In 3NF, Some Redundancies are Tolerated

Manager	Project	Branch
Brown	Mars	Chicago
Green	Jupiter	Birmingham
Green	Mars	Birmingham
Hoskins	Saturn	Birmingham
Hoskins	Venus	Birmingham

!!

Normalization -- 27

Decomposition into 3NF

- Decomposition into 3NF can proceed as follows.
  - ✓ For each functional dependency of the form $X \rightarrow Y$, where X contains a subset of a key K of r, create a projection on all the attributes X, Y (2NF).
  - ✓ For each dependency of the form $X \rightarrow Y$, where X, doesn't contain any key attributes, and not all attributes of Y are key attributes, create a projection on all the attributes X, Y (3NF).
- The new relations only include dependencies $X \rightarrow Y$, where X contains a key K of r, or Y contains only key attributes.

Normalization -- 28

A Revised Example

Manager	Project	Branch	Division
Brown	Mars	Chicago	1
Green	Jupiter	Birmingham	1
Green	Mars	Birmingham	1
Hoskins	Saturn	Birmingham	2
Hoskins	Venus	Birmingham	2

Functional dependencies:

- Manager $\longrightarrow$ Branch, Division -- each manager works at one branch and manages one division;
- Branch, Division $\longrightarrow$ Manager -- for each branch and division there is a single manager;
- Project, Branch $\longrightarrow$ Division -- for each branch, a project is allocated to a single division and has a sole manager responsible.

A Good Decomposition

Manager	Branch	Division
Brown	Chicago	1
Green	Birmingham	1
Hoskins	Birmingham	2

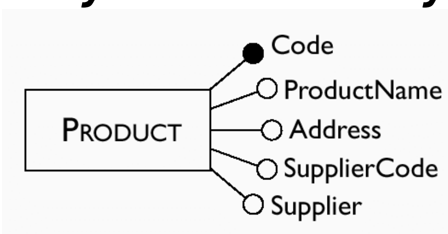
Project	Branch	Division
Mars	Chicago	1
Jupiter	Birmingham	1
Mars	Birmingham	1
Saturn	Birmingham	2
Venus	Birmingham	2

- The decomposition is in 3NF but not in BCNF; moreover, it is lossless and dependencies are preserved.
- This example demonstrates that Boyce–Codd normal form is too strong a condition to impose on a relational schema.

Database Design and Normalization

- The theory of normalization can be used as a basis for quality control operations on schemas, during both conceptual and logical design.
- Analysis of the relations obtained during the logical design phase can identify places where the conceptual design was inaccurate: such a validation of the design is usually relatively easy.
- Normalization can also be used during conceptual design for quality control of each element of a conceptual schema (entity or relationship).

Analysis of an Entity



- The functional dependency

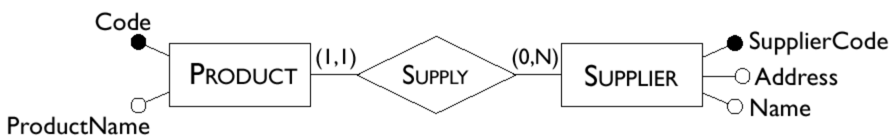
  SupplierCode → Supplier, Address

  is verified on the attributes of the entity: all the properties of each supplier are identified by its SupplierCode.
- The entity violates the third normal form since this dependency has a left hand side that does not contain the identifier and a right hand side made up of attributes that are not part of the key.

Decomposing Product

- Supplier is (or should be) an independent entity, with its own attributes (code, surname and address)
- If Product and Supplier are distinct entities, they should be linked through a releationship.
- Since there is a functioanal dependency from Code to SupplierCode, we are sure that each product has at most one supplier (maximum cardinality 1).
- Since there is no dependency from SupplierCode to Code, we have an unrestricted maximum cardinality (N) for Supplier in the relationship.

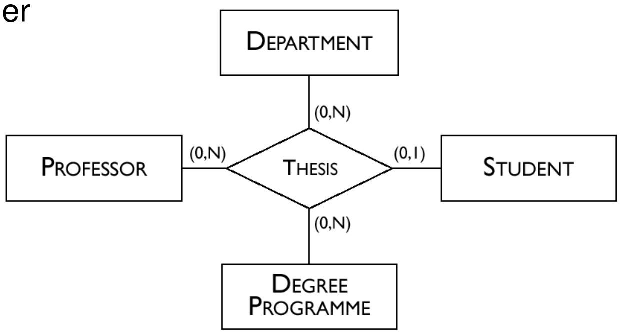
Decomposing Product



- This decomposition satisfies the two fundamental properties:
  - ✓ It is a lossless decomposition, because of the one-to-many relationship which allows us to recostruct the values of the attributes of the original entity;
  - ✓ Moreover, It preserves the dependencies, because each of the dependencies is embedded in one of the entities or can be reconstructed from them.

Analysis of a Relationship

- It is easy to check whether a binary relation is in 3NF (or in BCNF), the verification of normalization need only be carried out on n-ary relationships for n≥3, since binary relationships can't be decomposed.
- Consider



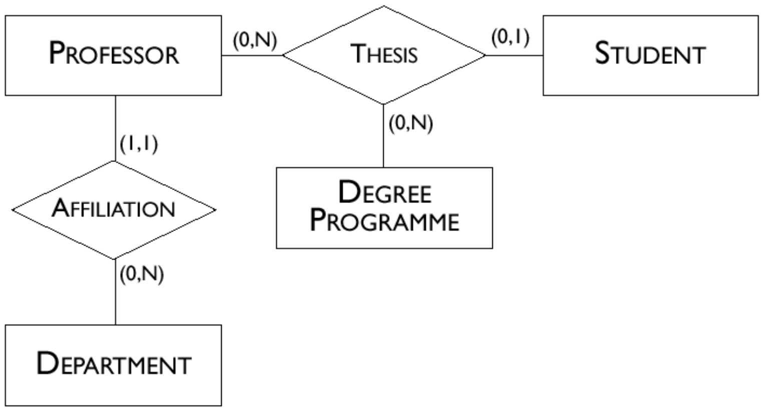
Some Functional Dependencies

- ✓ Student → DegreeProgramme (each student is enrolled in one degree programme)
- ✓ Student → Professor (each student writes a thesis under the supervision of a single professor)
- ✓ Professor → Department (each professor is associated with a single department and the students under her supervision are students in that department)

- The (unique) key of the relationship is Student (given a student, the degree programme, the professor and the department are identified uniquely)
- The third functional dependency causes a violation of 3NF.

Decomposing Thesis

- The following is a decomposition of Thesis where the two decomposed relationships are both in 3NF(also in BCNF)



More Observations...

- The relationship Thesis is in 3NF, because its key is made up of the Student entity, and its dependencies all have this entity on the left hand side.
- However, not all students write theses, therefore not all students have supervisors.
- From a normal form point of view, this is not a problem.
- However, our conceptual schema should reflect the fact that being in a degree programme and having a supervisor are independent facts.

Another Decomposition