Complex Systems Theory and Practice

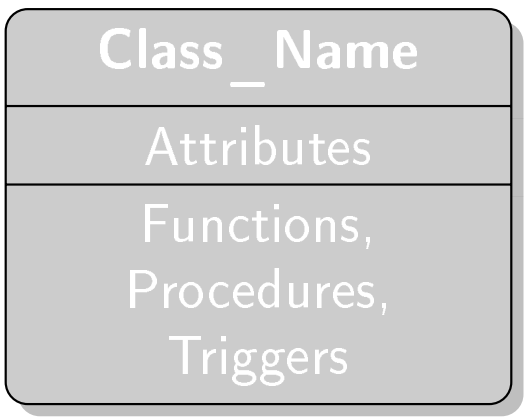
Lecture 6

Piotr Syga

2021-11-19

Class diagrams

- each entity is represented by a separate class
- we may specify the entity at different level of details (class names only, adding attributes, adding functions and triggers, . . .)
- we can create class trees (inheritance) to allow different levels of entity detail



Associations

- indicate class relationships
- association label indicates multiplicity and direction of association
- in the case of autoassociation, it is necessary to specify the roles and the multiplicity at each end of the association
- an association can have its own attributes (association classes, connected by a dashed line to an association)

Aggregation and composition

- stronger types of relationships (required part of a whole)
- as opposed to association, the existence of one entity can be conditioned by the existence of another
- we denote the attributes and multiplicities as in association
- we denote them in the diagram indicated by an empty and a filled diamond, respectively

Association modeling

- one-to-one: we extend the table or use one of the solutions below
- one-to-many: we implement a table for each entity and implement foreign keys (or use the solution for many-to-many relationship)
- many-to-many: we implement a separate table (dictionary) for the relationship – we create foreign keys to both connected tables

Specialization and inheritance

In relational databases, the inheritance typical of object-oriented databases and programming languages has to be modeled in a different way:

- A single table for the entire class tree with a details attribute that marks the specialization – mandatory comments
- Adding the attributes of the main class and the inheriting class, we create separate tables for all derived classes
- Separating each class, modeling inheritance relationships by aggregation/composition with fixed multiplicities

Modeling NoSQL databases

- Each document may have a different structure
- No fixed relationships between collections
- We model the base with the application in mind (its restrictions and possibilities)
- Documents are modeled as entities aggregated with collection entities
- We model the connections between entities taking into account programming language classes/interfaces

Requirements examples

Description: Design a database for a football league.

Requirements examples

Description: Design a database for a football league. Include players, teams, and games played.

Requirements examples

Description: Design a database for a football league. There is always at least one team in the league, each team consists of multiple players and at least one coach. Teams are identified by their name and their headquarters, whereas people are identified by their first name, last name and date of birth. Additionally, information about goals, assists and cards for each player is stored. (For each team, on the other hand, we know the . . . .)
Each match takes place in a specific stadium, within a set competition. It is played by the home team and the visiting team, an ordered pair of teams along with the season of the game uniquely identifies the game. No team can play a match two days in a row. For each match we store its score, goal scorers, information about assists, cards, the referee of the match and additional statistics such as percentage of ball possession (the sum of ball possession by both teams can not exceed 100). The database should automatically update the players' statistics based on the added match. It should be possible to generate the current standings. After all matches have been played (a match and a rematch between each pair of teams) the season should be automatically ended.

Tool examples for ERD and UML

- GitMind Online
- Dia
- NetBeans UML
- Oracle J/SQL Developer
- Eclipse UML
- UML Designer
- Ubrello
- UMLet
- Papyrus
- UML Modeler
- Vertabelo
- TikZ – example, TikZ-UML package

Idea

Main goals

- removing redundant data
- coping with the modification anomalies (adding new records, data lost during row deletion or update)

Simple example

ID	FName	LName	RoomNo	DeptID	FacID	DeptName	CID	CName
1	John	Doe	107	D11/2	11	Computer Science	I04	Data bases
2	Jane	Doe	107	D13/4	13	Mathematics	T06	Graph Theory
3	John	Smith	387	D4/1	4	Computer Science	Z17	System architecture
4	Jane	Smith	41	D11/6	11	Theoretical Physics	I23	Complex Systems

- how do we insert a new course name?
- what happens after deleting the row with ID=2?
- what happens after changing the last name in row with ID=2?

Functional dependence

Basic notions

Let U be a finite subset of all attributes of some relation R and X, $Y \subseteq U$. Then, if the values of attributes in X determine the values of attributes in Y we say that Y *depends functionally* on X. Note that in such case $\pi_{X,Y}(R)$ is a function. Additionally, if $Y \subseteq X$ we say that Functional dependence $X \to Y$ is a trivial one.

Functional dependence

Examples

- Postal Code $\rightarrow$ City
- Postal Code $\rightarrow$ City, Street
- City, Street $\rightarrow$ Postal Code
- FName, LName $\rightarrow$ DeptName
- DeptID $\rightarrow$ DeptName,FacID
- ID $\rightarrow$ R

Attribute closure

Definition

A *closure of an attribute* (or a set of attributes) X is a set of attributed that can be derived from X based on the functional dependencies $\mathcal{F}$.

Algorithm

1. If $i = 0$, $\{X\}_i^+ = \{X\}$, while $\{X\}_i^+$ changes
2. For each $(Y \to Z) \in \mathcal{F}$:
   - If $Y \subseteq \{X\}_i^+$, Add Z to $\{X\}_i^+$
   - $i \leftarrow i + 1$

Attribute closure

Examples:

$\{\text{DeptName}\}^+ = \{\text{DeptName, FName}\}^+ = \{\text{DeptName, FName}\}$
$\{\text{RoomNo}\}^+ = \{\text{RoomNo, LName}\}$
$\{\text{LName}\}^+ = \{\text{LName}\}$
$\{\text{LName, FName}\}^+ =$
$\{\text{LName, FName, RoomNo, DeptID, FacID, DeptName, CID, CName, ID}\}$

Keys

Idea

In order to uniquely identify rows, we need a (possibly small) set of attributes that determines all other columns. To find such a set we'll search the possible subsets of all the attributes and calculate their closure.
To avoid long notation of the table from previous slides let as assume that we have a table R with 7 columns that we denote $R = ABCDEFG$.
Moreover, we already identified the set of all functional dependencies $\mathcal{F} = \{A \to CD, B \to D, AD \to F, BD \to C, AEG \to BCFG, CG \to B, FG \to D, DE \to CEG\}$.
They key will be a subset $X \subseteq R$ such that $X^+ = R$.

Keys

Definitions

Let R be a relation with attributes' set U and a set of functional dependencies $\mathcal{F}$. A *key* of R is a set $K \subseteq U$, s.t.$\{K\}^+ = U$. Moreover, if there is no subset $P \subset K$ such that $\{P\}^+ = U$, then K is a *minimal key* (or a candidate key). One of the minimal keys is selected to be a primary key (it does not have to be the one with the fewest attributes).

Primary attributes

If K is a candidate key in R with set of attributes U, then all attributes $X \in K$ is a *key attribute* (also referred to as *primary attribute*). Let $\mathcal{K}$ be a set of all key attributes, then all attributes from $U \setminus \mathcal{K}$ are non-prime attributes.

Keys

Examples:

- obviously, for every R with attributes U, $K = U$ is a key
- for our table with courses we already shown that $\{\mathrm{LName}, \mathrm{FName}\}$ is a key and that LName is not (neither is FName), hence it is a candidate key (we can find many more)
- for R above we can 'guess' that $ABCDE$ is a key, since F depends on AD and G depends on DE. But is it a candidate key? Are there any more?

Learning the key

Finding keys for $R = (U, \mathcal{F})$ – extensive search

While there are unchecked subsets:

1. Take the first unchecked functional dependency from $\mathcal{F}$: $X \to Y$
2. Calculate $T = \{X\}^+$, if $T = U$ add X to the list of keys
3. While $T \subset U$, add $Y \in U \setminus T$ to our 'potential key' and find $T = \{XY\}^+$
4. Check surplus attributes in the keys you found, namely if key $K = WZ$ if there is a non-empty subset W s.t. $\{WZ\}^+ = \{Z\}^+$

Learning the key

Finding keys for $R = (U, \mathcal{F})$ – extensive search

While there are unchecked subsets:

1. Take the first unchecked functional dependency from $\mathcal{F}$: $X \to Y$
2. Calculate $T = \{X\}^+$, if $T = U$ add X to the list of keys
3. While $T \subset U$, add $Y \in U \setminus T$ to our 'potential key' and find $T = \{XY\}^+$
4. Check surplus attributes in the keys you found, namely if key $K = WZ$ if there is a non-empty subset W s.t. $\{WZ\}^+ = \{Z\}^+$

Useful tips:

- if an attribute X does not provide any information (there is no non-trivial f.d. $XY \to Z$) it will never be a part of any candidate key
- if an attribute X cannot be derived from any other attributes (there is no non-trivial f.d. $Y \to X$) it will be a part of every key

Example of key finding

$R = ABCDEFG$, $\mathcal{F} = \{A \to CD,\ B \to D,\ AD \to F,\ BD \to C,\ AEG \to BCFG,\ CG \to B,\ FG \to D,\ DE \to CEG\}$.

We check if A is a key: $A^+ = ACDF$.
The B is missing, hence we check $\{AB\}^+ = \{ABCDF\}$.

...
$\{ABE\}^+ = ABCDEFG$ ✓

Is it a minimal key? AB is not a key, so we have to check other subsets – E, AE i BE?

Example of key finding

$R = ABCDEFG$, $\mathcal{F} = \{A \to CD,\ B \to D,\ AD \to F,\ BD \to C,\ AEG \to BCFG,\ CG \to B,\ FG \to D,\ DE \to CEG\}$.

We check if A is a key: $A^+ = ACDF$.
The B is missing, hence we check $\{AB\}^+ = \{ABCDF\}$.

...

$\{ABE\}^+ = ABCDEFG$ ✓

Is it a minimal key? AB is not a key, so we have to check other subsets – E, AE i BE? $E^+ = E$ – not a key
$AE^+ = ACDE^+ = ACDEFG^+ = ABCDEFG = R$ ✓
$BE^+ = BDE^+ = BCDEG^+ = BCDEG \neq R$

Example of key finding

$R = ABCDEFG$, $\mathcal{F} = \{A \to CD,\ B \to D,\ AD \to F,\ BD \to C,\ AEG \to BCFG,\ CG \to B,\ FG \to D,\ DE \to CEG\}$.

We check if A is a key: $A^+ = ACDF$.
The B is missing, hence we check $\{AB\}^+ = \{ABCDF\}$.
...
$\{ABE\}^+ = ABCDEFG$ ✓

Is it a minimal key? AB is not a key, so we have to check other subsets – E, AE i BE? $E^+ = E$ – not a key
$AE^+ = ACDE^+ = ACDEFG^+ = ABCDEFG = R$ ✓
$BE^+ = BDE^+ = BCDEG^+ = BCDEG \neq R$ ..or we could check that A is never on the right side of f.d.

1NF

First normal form

Relation R is in 1NF **iff** every attribute is atomic and there are no duplicate rows (i.e., there is a key).

2NF

Second normal form

Relation R is in 2NF **iff** R is in 1NF and no non-prime attribute depends functionally on a proper subset of a candidate key.

It means that if A is a non-prime attribute and X is a candidate key in R, then every f.d. $Y \to A$, where $Y \subset X$ violates 2NF.

Normalization

1. We select a f.d. $X \to Y$ violating 2NF
2. We decompose R w.r.t. the f.d. $U_1 = \{X\}^+$, $U_2 = (U \setminus \{X\}^+) \cup X$.
3. We end when no f.d. violates 2NF.

3NF

Third normal form

Relation R is in 3NF **iff** R is in 2NF and all non-prime attributes depend only on a complete candidate key.

It means that for any non-prime attributes X, Y any f.d. $X \rightarrow Y$ violates 3NF.

3NF

Third normal form

Relation R is in 3NF **iff** R is in 2NF and all non-prime attributes depend only on a complete candidate key.

It means that for any non-prime attributes X, Y any f.d. $X \to Y$ violates 3NF.

In other words

Relation R is in 3NF **iff** for any f.d. $(X \to Y) \in \mathcal{F}$ for R, at least one condition is fulfilled:

- $Y \subseteq X$
- X is a key
- every attribute $A \in Y \setminus X$ is a primary attribute.

Decomposition of a relation

ID	FName	LName	RoomNo	DeptID
1	John	Doe	107	D11/2
2	Jane	Doe	107	D13/4
3	John	Smith	387	D4/1
4	Jane	Smith	41	D11/6

DeptID	FacID	DeptName
D11/2	11	Computer Science
D13/4	13	Mathematics
D4/1	4	Computer Science
D11/6	11	Theoretical Physics

Lecturer	CID	CName
1	I04	Data bases
2	T06	Graph Theory
3	Z17	System architecture
4	I23	Complex Systems

A Simple Guide to 5 Normal Forms in Relational DB Theory. W. Kent

Every non-key attribute must provide
a fact about the key (1NF),
the whole key (2NF),
and nothing but the key (3NF).
(So help me Codd.)

BCNF

Boyce–Codd Normal Form

Relation R is in BCNF **iff** for every f.d. $(X \to Y) \in \mathcal{F}$ of R at least one condition is fulfilled:

- $Y \subseteq X$
- X is a key

Note that **not every relation** may be normalized to BCNF without the loss of information (attributes or functional dependencies). Minimal (not)working example is a relation $R = ABC$ with f.d. $\{AB \to C,\ C \to A\}$.