Database Systems

Database Design

H. Turgut Uyar Şule Öğüdücü

2002-2015

License

Topics

Functional Dependency

- let Z be the set of all attributes of the relation R
- let $A, B \subseteq Z$
- A functionally determines B: $A \rightarrow B$
  for every A value there can only be one B value
- every functional dependency is an integrity constraint

Example Relation

R

MOVIEID	TITLE	COU	LANG	ACTORID	NAME	ORD
6	The Usual Suspects	UK	EN	308	Gabriel Byrne	2
228	Ed Wood	US	EN	26	Johnny Depp	1
70	Being John Malkovich	US	EN	282	Cameron Diaz	2
1512	Suspiria	IT	IT	745	Udo Kier	9
70	Being John Malkovich	US	EN	503	John Malkovich	14

- assumption: the language of the movie is the language of the country where it was made

Functional Dependency Examples

- MOVIEID $\rightarrow$ TITLE
- MOVIEID $\rightarrow$ {TITLE, COUNTRY, LANGUAGE}
- ACTORID $\rightarrow$ NAME
- {MOVIEID, ACTORID} $\rightarrow$ ORD
- trivial: MOVIEID $\rightarrow$ MOVIEID
- redundant: {MOVIEID, ACTORID} $\rightarrow$ COUNTRY

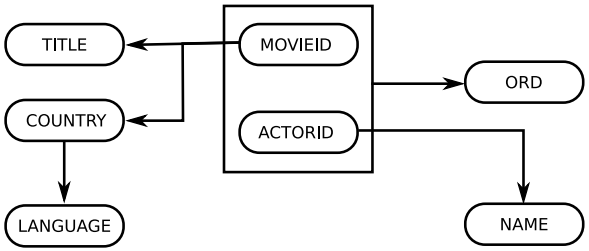
Irreducible Set

- let S be the set of all FDs of the relation
- let $T \subseteq S$
- T is an irreducible set of FDs if:
- T contains as few elements as possible, and
- every FD in S can be derived from the FDs in T
- let there be only one attribute on the right hand side of FDs

Irreducible Set Example

- MOVIEID $\rightarrow$ TITLE
- MOVIEID $\rightarrow$ COUNTRY
- COUNTRY $\rightarrow$ LANGUAGE
- ACTORID $\rightarrow$ NAME
- {MOVIEID, ACTORID} $\rightarrow$ ORD

Dependency Example



Normal Forms

- normal form: set of conditions that relations must satisfy
- 1NF, 2NF, 3NF, BCNF, 4NF, 5NF
- every form adds stricter conditions to the previous form
- NF of database = NF of weakest relation
- 1NF: attribute values have to be atomic
- simplifying assumption for definitions:
  A TABLE HAS ONLY ONE CANDIDATE KEY,
  WHICH IS ALSO THE PRIMARY KEY.

Normalization

- normalization:
  transforming relations from one form to the next, stricter form
- transformation must be lossless

Theorem (Heath)

- *let Z be the set of all attributes of the relation R*
- *let $A, B, C \subseteq Z$*
- $A \rightarrow B \Rightarrow R = (A, B)$ *join* (A, C)

Lossless Transition Example

R1

MOVIEID	TITLE	COU	LANG
6	The Usual Suspects	UK	EN
228	Ed Wood	US	EN
70	Being John Malkovich	US	EN
1512	Suspiria	IT	IT

R2

MOVIEID	ACTORID	NAME	ORD
6	308	Gabriel Byrne	2
228	26	Johnny Depp	1
70	282	Cameron Diaz	2
1512	745	Udo Kier	9
70	503	John Malkovich	14

- $R = R1$ *join* $R2$

Lossy Transition Example

R1

MOVIEID	TITLE	COU	LANG
6	The Usual Suspects	UK	EN
228	Ed Wood	US	EN
70	Being John Malkovich	US	EN
1512	Suspiria	IT	IT

R2

COU	ACTORID	NAME	ORD
UK	308	Gabriel Byrne	2
US	26	Johnny Depp	1
US	282	Cameron Diaz	2
IT	745	Udo Kier	9
US	503	John Malkovich	14

- $R \neq R1$ *join* $R2$
- {MOVIEID, ACTORID} $\rightarrow$ ORD

Anomalies

- *insert*: data is known but can not be inserted
- *delete*: deleting some data causes some other data to be lost
- *update*: updating data requires modifications in multiple tuples

Anomaly Examples

example database

- the country of the movie "Gattaca" is known to be US, but this cannot be inserted if there is no actor in the movie
- deleting that Johnny Depp acts in the movie "Ed Wood" also deletes that the movie was made in the US
- changing the country of the movie "Being John Malkovich" requires modifications in two tuples

2nd Normal Form

- 2NF: every non-key attribute depends on the primary key
- in a relation R that conforms to 1NF, if:
- $R(A, B, C, D)$, primary key: (A, B), and $A \rightarrow D$
- to transform to 2NF, divide into:
- $R1(A, D)$, primary key: A, and $R2(A, B, C)$, primary key: (A, B), where A is a foreign key referencing $R1$

1NF-2NF Transition Example

- among non-key attributes, only ORD depends on primary key
- A: {MOVIEID}
  B: {ACTORID}
  C: {NAME, ORD}
  D: {TITLE, COUNTRY, LANGUAGE}
- R1(MOVIEID, TITLE, COUNTRY, LANGUAGE)
  primary key: MOVIEID
- R2(MOVIEID, ACTORID, NAME, ORD)
  primary key: {MOVIEID, ACTORID}
  MOVIEID is a foreign key referencing R1

1NF-2NF Transition Example

- R2 still not 2NF: ACTORID → NAME
- A: {ACTORID}
  B: {MOVIEID}
  C: {ORD}
  D: {NAME}
- R3(ACTORID, NAME)
  primary key: ACTORID
- R4(MOVIEID, ACTORID, ORD)
  primary key: {MOVIEID, ACTORID}
  ACTORID is a foreign key referencing R3

2NF Relation Examples

R1

MOVIEID	TITLE	COU	LANG
6	The Usual Suspects	UK	EN
228	Ed Wood	US	EN
70	Being John Malkovich	US	EN
1512	Suspiria	IT	IT

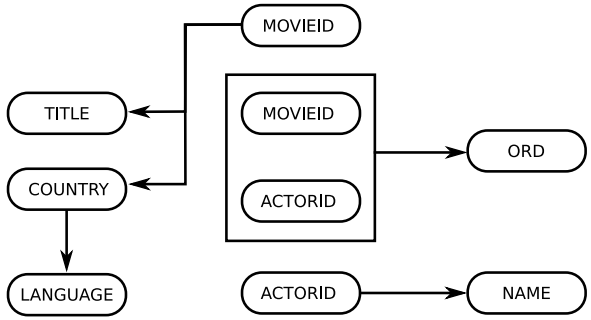
R3

ACTORID	NAME
308	Gabriel Byrne
26	Johnny Depp
282	Cameron Diaz
745	Udo Kier
503	John Malkovich

R4

MOVIEID	ACTORID	ORD
6	308	2
228	26	1
70	282	2
1512	745	9
70	503	14

Dependency Example



2NF Corrected Anomalies

example database

- if the country of the movie "Gattaca" is known to be US, this can be inserted to R1
- if Johnny Depp is deleted from the movie "Ed Wood", the country of the movie is still kept in R1
- changing the country of the movie "Being John Malkovich" requires updating only one tuple in R1

2NF Remaining Anomalies

example database

- it is known that movies made in Brazil are in Portuguese but this can not be inserted if there is no movie made in Brazil
- deleting the movie "Suspiria" also deletes that the language of movies made in Italy is Italian
- changing the language of the movies made in the US requires two tuples to be updated

3rd Normal Form

- 3NF: non-key attributes do not depend on any attributes other than the primary key
- in a relation R that conforms to 2NF, if:
- $R(A, B, C, D)$, primary key: A, and $C \rightarrow D$
- to transform to 3NF, divide into:
- $R1(C, D)$, primary key: C, and $R2(A, B, C)$, primary key: A, where C is a foreign key referencing $R1$

2NF-3NF Transition Example

- R1: COUNTRY $\rightarrow$ LANGUAGE
- A: {MOVIEID}
  B: {TITLE}
  C: {COUNTRY}
  D: {LANGUAGE}
- R5(COUNTRY, LANGUAGE)
  primary key: COUNTRY
- R6(MOVIEID, TITLE, COUNTRY)
  primary key: MOVIEID
  COUNTRY is a foreign key referencing R5

3NF Relation Examples

R6

MOVIEID	TITLE	COU
6	The Usual Suspects	UK
228	Ed Wood	US
70	Being John Malkovich	US
1512	Suspiria	IT

R5

COU	LANG
UK	EN
US	EN
IT	IT

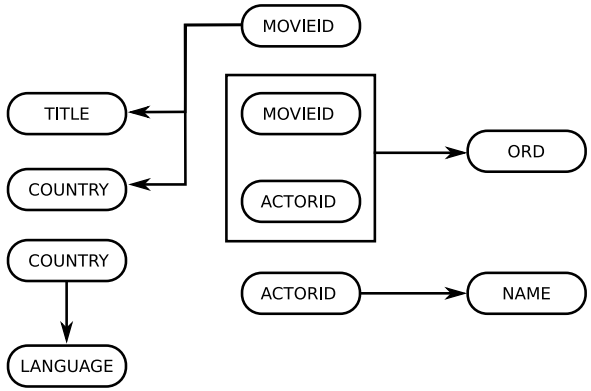
R3

ACTORID	NAME
308	Gabriel Byrne
26	Johnny Depp
282	Cameron Diaz
745	Udo Kier
503	John Malkovich

R4

MOVIEID	ACTORID	ORD
6	308	2
228	26	1
70	282	2
1512	745	9
70	503	14

Dependency Example



3NF Corrected Anomalies

example database

- if movies made in Brazil are in known to be in Portuguese, this can be inserted into R5
- if the movie "Suspiria" is deleted, R5 still keeps that movies made in Italy are in Italian
- changing the language of the movies made in the US requires modifying only one tuple in R5

Boyce-Codd Normal Form

- BCNF: all functional dependencies must be on candidate keys
- consider dependencies between key attributes

BCNF Example

example database

- let movie titles be unique
- candidate keys:
  {MOVIEID, ACTORID}
  {TITLE, ACTORID}
- non-conforming functional dependencies:
  MOVIEID → TITLE
  TITLE → MOVIEID

Views

- presenting a derived table like a base table: view
- isolating users and application programs
  from changes in database structure
- creating a view:

```
CREATE VIEW view_name AS
  SELECT ...
```

- SELECT will be executed every time the view is used

View Example

- identifiers, titles and years of new movies

```
CREATE VIEW NEW_MOVIE AS
  SELECT ID, TITLE, YR FROM MOVIE
    WHERE (YR > 1995)

SELECT * FROM NEW_MOVIE
```


Updating Views

- any change will have to performed on the base tables
- rules need to be defined
- creating a rule:

```
CREATE RULE rule_name AS
  ON event TO view_name
  [ WHERE condition ]
  DO [ INSTEAD ] sql_statement
```

View Rule Example

- modify the title of a new movie

```
UPDATE NEW_MOVIE SET TITLE = ...
  WHERE (ID = ...)
```

- rule for updating the base table

```
CREATE RULE UPDATE_TITLE AS
  ON UPDATE TO NEW_MOVIE
  DO INSTEAD
    UPDATE MOVIE SET TITLE = new.TITLE
      WHERE (ID = old.ID)
```


References

Required Reading: Date

- Chapter 11: Functional Dependencies
- Chapter 12: Further Normalization I: 1NF, 2NF, 3NF, BCNF
- Chapter 10: Views

Entity/Relationship Model

- modeling approach (Chen 1976)
- entities
- properties
- relationships

Entities

- entity: set of "things" with the same attributes
- elements of the set are *instances* of the entity
- *strong*: can exist by itself
- *weak*: existence depends on other entities

Entity Examples

- entity: movie, person
- person instance: Johnny Depp
- strong entity: person
- weak entity: movie

Properties

- property: data describing entity
- simple / composite
- key
- single / multiple valued
- empty
- base / derived

Property Examples

- property: title, country, language
- simple: first name, last name
- composite: full name
- base: date of birth
- derived: age

Relationships

- relationship: connection between entities
- *participant*: entities in the relationship
- *degree*: number of participants
- *total*: all instances of the entity participate in the relationship (otherwise *partial*)

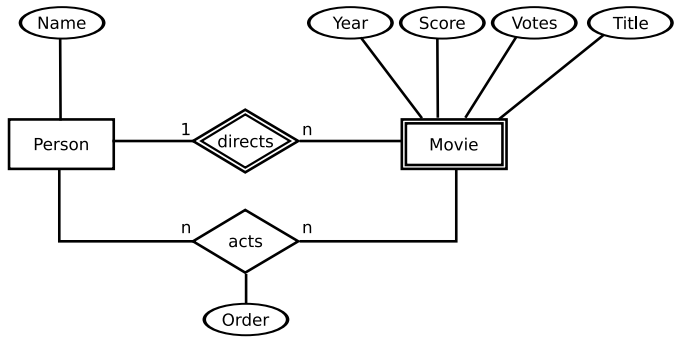
Relationship Types

- one-to-one
- e.g. capital city relationship between countries and cities
- one-to-many
- e.g. management relationship between employees and projects
- many-to-many
- e.g. enrollment relationship between students and courses

Entity/Relationship Diagrams

- entity: box
- weak: double lines
- property: circle
- composite: sub-circles
- relationship: diamond
- between weak and strong: double lines
- total: connection double lines
- 1 or n depending on the type of the relationship

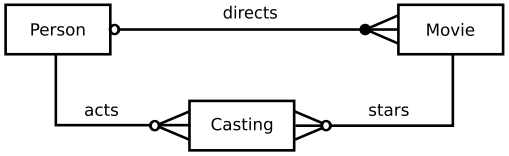
Entity/Relationship Diagram Example



Entity/Relationship Diagrams

- properties excluded if diagram too large
- alternative representation for relationship: line
- name of relationship above the line
- fork at the end of line if “many”
- circle at the end of line: empty if partial, filled if total
- no many-to-many relationships
- add an entity and use two one-to-many relationships

Entity/Relationship Diagram Example



Applying to Design

- every entity is a relation
- every property is an attribute
- every one-to-many relationship is a foreign key from the "many" side to the "one" side
- every many-to-many relationship is a relation with foreign keys to participating entities

References

Required Reading: Date

- Chapter 14: Semantic Modeling