Relational database design

Sections to read:

- “Functional dependencies”
- “Rules about functional dependencies”

Revision of the relational model

- Define relation schemas
  - Define tables using SQL CREATE TABLE statements
- Integrity constraints
  - primary keys and foreign keys
- An example relational database schema :

  Movie(<u>mvID</u>, title)

  Actor(<u>name</u>, DOB, birth-country)

  Cast(<u>mvID*, actor-name*</u>)

Relational Database Design

- Designing a relational database is about designing the schemas for relations (tables) --- how to organize attributes (data) into relations.
  - Design an ER diagram and then map it into a relational database schema.
  - Design relation schemas directly from requirements.

The Movies Database -- Relational Database Schema …

- Is the previous village cinema database schema a good schema for keeping data?
- How to justify the design?

Solution:

- Functional dependencies
- Normal forms.

The Functional Dependency (FD)

- The functional dependency generalises the concept of keys for relations:
  - A key of a relation is a rule constraining data to keep in the relation (table): when any two tuples agree on the values for attributes in a key, the two tuples agree with each other.
  - A key determines a relation (relation schema).
  - A functional dependency on a relation is a rule for keeping data in a relation: when any two tuples agree on some attributes, they agree on some other attributes.

The FD …

Let $\{x_1, x_2, \ldots, x_n\}$ represent a set of n attributes, and y represents an attribute, a functional dependency

$$x_1, x_2, \ldots, x_n \rightarrow y$$

represents a rule for keeping data in a relation: in a relation that contain attributes $x_1, x_2, \ldots, x_n, y$, whenever two tuples have the same value for $x_1, x_2, \ldots, x_n$, they must have the same value for y.

FD …

- Generally an FD $X \rightarrow Y$ denotes relationship between two sets of attributes

$X=\{x_1, x_2, \ldots, x_n\}$, $Y=\{y_1, y_2, \ldots, y_m\}$.

Or equivalently,

$x_1, x_2, \ldots, x_n \rightarrow y_1$

$x_1, x_2, \ldots, x_n \rightarrow y_2$

…

$x_1, x_2, \ldots, x_n \rightarrow y_m$

$X \rightarrow Y$:
All attributes together in X determine any attribute in Y.

Example: FDs

In the mini-world of course management at RMIT

- Courses have course No (cno) and title. Each course has a unique *cno* – in a relation containing *cno* and *title*, whenever two courses agree on *cno*, they must agree on course title.
  - FD 1:cno → title
- Each room is designed for one type of classes, lecture or tutorial. Whenever two classes agree on room, they must also agree on type (class type).
  - FD 2: room → type
- Two courses (of the same content) may be scheduled into one class (mixed lectures for isys1055 and isys1057).
  - ~~day, time, room → cno~~ ✗

Example: Sample data

cno → title

room → type

cno	title	day	time	room	type
isys1057	Database Concepts	Wed	1030	14.04.27	lect
isys1057	Database Concepts	Thur	1330	14.04.27	lect
isys1055	Database Concepts	Wed	1730	12.05.02	lect
isys1057	Database Concepts	Wed	1130	14.10.30	tute
isys1057	Database Concepts	Wed	1330	14.09.23	tute
acct1009	Another Course	Wed	1130	14.04.27	lect
acct1009	Another Course	Thur	1330	07.02.23	lect
acct1009	Another Course	Thur	1430	07.02.23	lect

Where do FDs come from?

- FDs are natural constraints on data from business rules in the real world.
  - FDs do not depend on any sample data.
  - FDs are basic constraints at the attribute level. FDs constrain how data is related inside relations no matter how attributes are grouped into relations.
- Example: from the partial listing of Class data below it may seem

  room → cno ✗

  but it is not true.

cno	title	day	time	room	type
isys1057	Database Concepts	Wed	1030	14.04.27	lect
isys1057	Database Concepts	Thur	1330	14.04.27	lect
isys1055	Database Concepts	Wed	1730	12.05.02	lect
isys1057	Database Concepts	Wed	1130	14.10.30	tute
isys1057	Database Concepts	Wed	1330	14.09.23	tute

Inference rules for FDs

Armstrong's axioms:

- Reflexivity (trivial FDs)
- Transitivity
- Augmentation

The axioms allow reasoning about FDs – infer new FDs from existing FDs.
See Section 3.2 of the textbook.

Trivial FDs

- FDs that are always true following the definition but do not express real-world constraints are trivial FDs.
- An FD $X \rightarrow Y$ is a trivial FD if X includes Y (Y is a subset of X).
- Example:
  - cno → cno
  - cno, title → cno
  - day, time, room → day, time

Inference of FDs: Transitivity

- New FDs can be inferred from existing FDs.
- For example, given

  room → type

  type → length

It can be inferred that, *transitively*,

  room → length

This rule is called the transitive rule, or transitivity. This rule is intuitively true. (See Section 3.2.6 of the text for a formal proof)

Inference of FDs – Augmentation

Given

room → type

it can be inferred that, new FDs by augmenting the left and/or right sides of the FD with some attributes also hold:

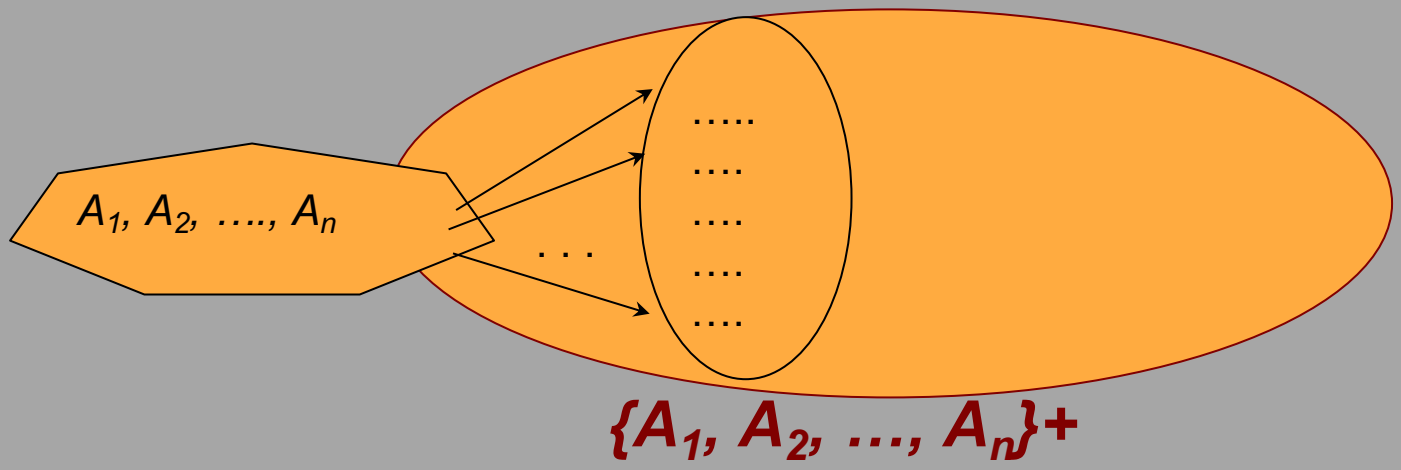
room, cno → type, cno

The closure of an attribute set

- Given a set of attributes *{A1, ..., An}* and a set of FDs *S*, the closure of *{A1, ..., An}*+ under *S* is the set of attributes B such that $A_1, ..., A_n \rightarrow B$ can be inferred from the FDs in *S*.

Compute the Closure for a set of Attributes

- Given a set of attributes $\{A_1, ..., A_n\}$ and a set of FDs S, compute the closure $\{A_1, ..., A_n\}^+$.
  1. Initialise $\{A_1, ..., A_n\}^+$ to $\{A_1, ..., A_n\}$.
  2. Apply FDs to determine new attributes from attributes in the current closure. Add the new attributes to the closure.
  3. Repeat step 2 until no more attributes can be added to the closure.
- A pseudo code algorithm and proof of its correctness is given in Sections 3.2.4 and 3.2.5 of the textbook.



Example: the closure for a set of attributes

- Given

Class(cno, title, day, time, room, type}

  - room → type
  - cno → title
- What is {day, time, room, cno}$^+$?
  - {day, time, room, cno, title, type} – all attributes in relation Class.
- What is {cno}$^+$?
  - {cno, title}

Exercise

Consider the following attributes to keep data for the village cinema database:

mvID, Title, Rating, Rel_date,Length,

Genre-name, studio-name

- What are the likely FDs (according to the real world situation)?
  - mvid → title
  - mvid → length
  - mvid → studio-name (Is this true?)
- What is $\{mvID\}^+$?

Minimal Basis for FDs

- Redundant FDs: FDs that can be inferred from other FDs.
- Redundant attributes in FDs: some attributes on the left hand side of FDs can be removed.
- The minimal basis for FDs is equivalent to the FDs but without redundancy:
  - Right sides are single attributes.
  - No FD can be removed.
  - No attribute can be removed from left hand side.

Finding a Minimal Basis for FDs

1. Replace an FD with multiple attributes on the right with FDs with single attribute on the right.
2. Remove any redundant FD.
3. Remove any redundant attribute from the left side of an FD.

Example

Consider FDs:

cno → title, prog, progtype
prog → progtype, progleader
day, time, room → classtype
room → classtype, capacity

1. Replace FDs with FDs with single attribute on the right hand side:

cno → title
cno → prog
~~cno → progtype~~ (redundant FD)
prog → progtype
prog → progleader
~~day, time, room → classtype~~ (redundant attribute on the left)
room → classtype
room → capacity

2. Redundant FDs are removed.
3. Redundant attributes on the left hand side are removed.

Example …

. The minimal basis for FDs:

cno → title
cno → prog
prog → progtype
prog → progleader
room → classtype
room → capacity

Key redefined

Given a set of FDs on relation *R*, a set of attributes *K* is a key for relation *R* if

- $K \rightarrow R$ (or equivalently $K^+ = R$)
- *K* is minimal: there is not a subset *K' of K* such that $K' \rightarrow R$.

Note that *R* is the set of all attributes in the relation.

- $K \rightarrow R$ is equivalent to $K \rightarrow R - K$. *$R - K$ is all the attributes of R other than K.*

Key …

- A relation can have several (candidate) keys, of which one is specified as the primary key.
- We have not included the concept of the superkey in the textbook for ease of understanding.
  - A superset of a key is called a superkey.

Key …

Given two FDs:

- cno → title
- room → type

What are the (candidate) keys for the below relation?

Class(cno, title, day, time, room, type)

{day, time, room, cno} is a minimum set of attributes that determine the whole relation and is therefore a key.

Key …

cno → title; room → type

Class(cno, title, day, time, room, type)

- Any subset of {day, time, room, cno} does not determine relation Class.

  day, room → cno ✗

  day, time → type ✗

  time, room → cno ✗

- Any superset of {day, time, room, cno} does not minimally determines relation Class.

  day, time, room, cno, type → cno, title (redundant attributes on the left)

- {day, time, room,cno} is the only (candidate) key; no other candidate keys for Class.

Bad design leads to data redundancy

Class(cno, title, day, time, room, type)

cno	title	day	time	room	type
isys1057	Database Concepts	Wed	1030	14.04.27	lect
isys1057	Database Concepts	Thur	1330	14.04.27	lect
isys1055	Database Concepts	Wed	1730	12.05.02	lect
isys1057	Database Concepts	Wed	1130	14.10.30	tute
isys1057	Database Concepts	Wed	1330	14.09.23	tute
acct1009	Another Course	Wed	1130	14.04.27	lect
acct1009	Another Course	Thur	1330	07.02.23	lect
acct1009	Another Course	Thur	1430	07.02.23	lect

The sample data has data redundancy.

Bad design leads to insertion anomaly

cno	title	day	time	room	type
isys1057	Database Concepts	Wed	1030	14.04.27	lect
isys1057	Database Concepts	Thur	1330	14.04.27	lect
isys1055	Database Concepts	Wed	1730	12.05.02	lect
isys1057	Database Concepts	Wed	1130	14.10.30	tute
isys1057	Database Concepts	Wed	1330	14.09.23	tute
acct1009	Another Course	Wed	1130	14.04.27	lect
acct1009	Another Course	Thur	1330	07.02.23	lect
acct1009	Another Course	Thur	1430	07.02.23	lect

- Insertion anomaly: When insertion only occurs in one place, data inconsistency occurs.
- Example: There is a new tutorial class for isys1057 scheduled at Wed 10:30 in 14.10.02. If we only insert cno but not title information, data inconsistency occurs.

Bad design leads to update anomaly

cno	title	day	time	room	type
isys1057	Database Concepts	Wed	1030	14.04.27	lect
isys1057	Database Concepts	Thur	1330	14.04.27	lect
isys1055	Database Concepts	Wed	1730	12.05.02	lect
isys1057	Database Concepts	Wed	1130	14.10.30	tute
isys1057	Database Concepts	Wed	1330	14.09.23	tute
acct1009	Another Course	Wed	1130	14.04.27	lect
acct1009	Another Course	Thur	1330	07.02.23	lect
acct1009	Another Course	Thur	1430	07.02.23	lect

- Update anomaly: When update only happens in one place but not all places, data inconsistency occurs.
- Example: If the course isys1057 is renamed to "Database Fundamentals" during program renewal, if we only change one place, inconsistency occurs.

Bad design leads to deletion anomaly

cno	title	day	time	room	type
isys1057	Database Concepts	Wed	1030	14.04.27	lect
isys1057	Database Concepts	Thur	1330	14.04.27	lect
isys1055	Database Concepts	Wed	1730	12.05.02	lect
isys1057	Database Concepts	Wed	1130	14.10.30	tute
isys1057	Database Concepts	Wed	1330	14.09.23	tute
acct1009	Another Course	Wed	1130	14.04.27	lect
acct1009	Another Course	Thur	1330	07.02.23	lect
acct1009	Another Course	Thur	1430	07.02.23	lect

- Deletion anomaly: Deletion of one piece of information results in deletion of other information as a side effect.
- Example: If the last scheduled class for acct1009 is deleted from the timetable this semester, the course acct1009 is complete lost, together with its title information.

Solution

- Measure the goodness of a relation schema design in normal forms. Higher normal forms remove more data redundancy. BCNF is free from data redundancy.
- 1NF (first normal form): a relation is in 1NF if attributes take atomic values (not a set or structure value).
- 2NF (second normal form): a relation in 1NF and non-key attributes are fully functional dependent on a key.
- BCNF (Boyce-Codd normal form): a relation R is in BCNF if for any FD $X \rightarrow Y$ holds in R and Y contains only attributes not in X, X is a key of R.

Example

Assume Class(cno, title, day, time, room, type) in 1NF and FDs:

cno → title

room → type

- The only key is {day, time, room, cno}.
- cno → title violates 2NF (and BCNF), as title is partially dependent on the key.
- room → type violates 2NF (and BCNF) as type is partially dependent on the key.
- Class is not in 2NF, and not in BCNF either.

Exercise

Consider the relation

BookDetails(isbn, title, publisher, publisher_addr, author),

and the following business rules:

- A book has a unique ISBN, and a title.
- A book has one publisher. A publisher has an address.
- A book can have more than one author.

- What are the likely FDs?
- What are the candidate keys for relation BookDetails?
- Is relation BookDetails in BCNF?

Exercise: The Movies Database Schema

- Discuss likely FDs for each relation.
- Use FDs to examine the correctness of primary key annotations for each relation.

Is the database schema a good design? In other words, is the database in BCNF?

Movie(mvID, Title)
Director(director-name, DOB, birth-country)
Actor(actor-name, DOB, birth-country)
Genre(genre-name, description)
Novel(novel-title, author)
Classification(mvID*, genre-name*, note)
Cast(mvID*, actor-name*)
Direct(mvID*, director-name*)
Adaptation(mvID*, novel-title*)

Summary

- To evaluate relation schemas, normal forms are defined using FDs, including 1NF (first normal form), 2NF(second normal form), and BCNF (Boyce-Codd normal form).
- Normal forms are increasingly stricter: relations in BCNF are also in 2NF and relations in 2NF are in 1NF.
- Objective of relational database design: all relation schemas are in BCNF and free from data redundancy or update or deletion anomalies.