STA261
MANAJEMEN DATA RELASIONAL

Design Guidelines for Relational DBs

DEPARTEMEN STATISTIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT PERTANIAN BOGOR

SEMESTER GANJIL 2021/2022

Introduction

- Each **relation schema** consists of a number of **attributes**, and the **relational database schema** consists of a number of **relation schemas**
- Some formal way of analyzing why one **grouping of attributes** into a relation schema may be better than another
- Measure the **quality** of the design, other than the **intuition** of the designer
- Database design may be performed using two approaches: **bottom-up** (design by **synthesis**) or **top-down** (design by **analysis**).

Introduction

- A **bottom-up design** methodology considers the basic relationships *among* ***individual attributes*** as the starting point and uses those to construct relation schemas
- In contrast, a **top-down design** methodology starts with a number of ***groupings of attributes*** into relations that exist together naturally, for example, an invoice, a form, a report, or a transcript.

Introduction

- The implicit goals of the design activity are **information preservation** and **minimum redundancy**
- **Information** is very hard to quantify—maintaining all concepts, including attribute types, entity types, and relationship types
- **Minimizing redundancy** implies minimizing redundant storage of the same information and reducing the need for multiple updates to maintain consistency of the same information in response to update

Motivation

empID	name	job	deptID	dept
1	Bob	Programmer	1	Engineering
2	Alice	DBA	2	Databases
3	Kim	Programmer	1	Engineering

Case 1: Try adding a new employee in Engineering Dept, do you feel uneasy?
[**Insertion anomaly**, must repeat both deptID and dept (= dept name)]

Case 2: Try deleting the employee Alice, do you feel uneasy?
[**Deletion anomaly**, lose the existence of the Databases dept]

Case 3: Try changing the name of Engineering department into Technology, do you feel uncomfortable?
[**Update anomaly**, must change name in multiple rows]

Design Guidelines for Relational DBs

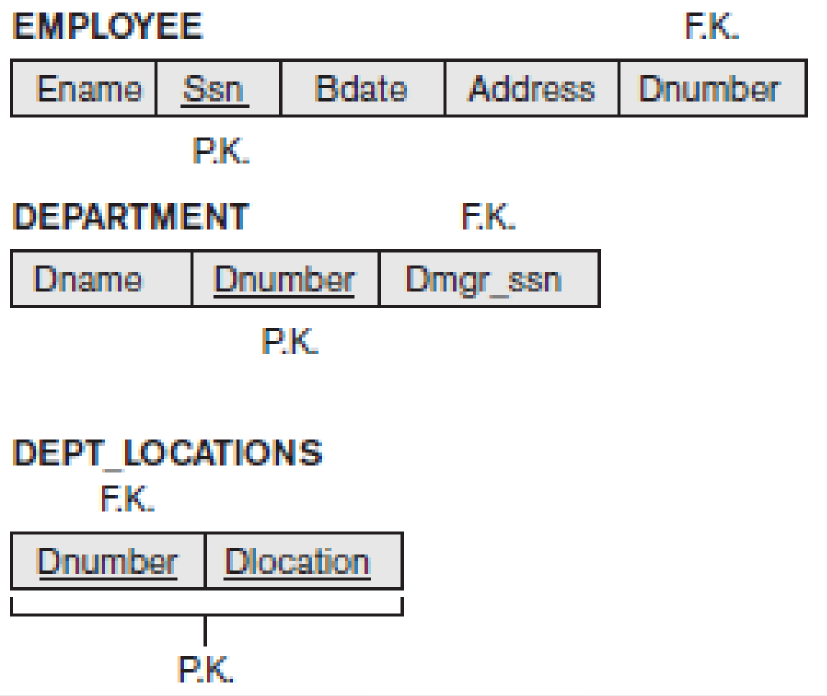
- DB schema can be seen in two levels:
  - User (***logical level***): how users interpret relational schema and its attributes → base and virtual relations
  - ***Storage level***: how tuples are stored and updated → base relations
- We focus here on base relations
  - What are the criteria for good *base relations*?

Design Guidelines for Relation Schemas

- Semantics of relational attributes
  - ✓ Making sure that the semantics of the attributes is clear in the schema
- Duplicated data and update anomaly
  - ✓ Reducing the redundant information in tuples
- Nulls
  - ✓ Reducing the NULL values in tuples
- Spurious tuples
  - ✓ Disallowing the possibility of generating spurious tuples

Semantics on Relational AttributesCreating a Schema

- Example:



- ➢ Each tuple represents an EMPLOYEE, with values for the employee's name (**Ename**), Social Security number (**Ssn**), birth date (**Bdate**), and address (**Address**), and the number of the department that the employee works for (**Dnumber**).
- ➢ The **Dnumber** attribute is a **foreign key** that represents an *implicit relationship* between EMPLOYEE and DEPARTMENT.
- ➢ Each DEPARTMENT tuple represents a department entity, and each PROJECT tuple represents a project entity.
- ➢ The schema DEPT_LOCATIONS represents a **multivalued** attribute of DEPARTMENT.

Semantics on Relational Attributes Creating a Schema

- Example:

SSN	Name	BirthData	Sex	Address	DNo	DName	DMgrSSN

SSN	Name	BirthData	Sex	Address	DNo

DNo	DName	DMgrSSN

Duplicated data and update anomaly

- Example:

  A poorly designed database causes ***anomalies***:

Student	Course	Room
Mary	CS145	B01
Joe	CS145	B01
Sam	CS145	B01
..	..	..

If every course is in only one room, contains ***redundant*** information!

Duplicated data and update anomaly

- Example:

  A poorly designed database causes ***anomalies***:

Student	Course	Room
Mary	CS145	B01
Joe	CS145	C12
Sam	CS145	B01
..	..	..

If we update the room number for one tuple, we get inconsistent data → **update anomaly**

Duplicated data and update anomaly

- Example:

  A poorly designed database causes ***anomalies***:

Student	Course	Room
Mary	CS145	B01
Joe	CS145	B01
Sam	CS145	B01
..	..	..

If every one drops the class, we lose what room the class is in → **delete anomaly**

Duplicated data and update anomaly

- Example:

  A poorly designed database causes ***anomalies***:

Student	Course	Room
Mary	CS145	B01
Joe	CS145	B01
Sam	CS145	B01
..	..	..

...	CS229	C12

Similarly, we can't have a room without students
→ ***insert* anomaly**

Duplicated data and update anomaly

- Example:

A poorly designed relation

Student	Course	Room
Mary	CS145	B01
Joe	CS145	B01
Sam	CS145	B01
..	..	..

The Better Form:

Student	Course
Mary	CS145
Joe	CS145
Sam	CS145
..	..

Course	Room
CS145	B01
CS229	C12

We will learn a theory to understand why this design may be better **and** how to find this *decomposition*…

Duplicated data and update anomaly

- Example:

Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321

Redundancy Redundancy

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland

Duplicated data and update anomaly

Exercise:

What can we improve to the following relation?

empID	name	job	deptID	dept
1	Bob	Programmer	1	Engineering
2	Alice	DBA	2	Databases
3	Kim	Programmer	1	Engineering

Duplicated data and update anomaly

Exercise:

gameRelease

	Developer	Game	Platform	ReleaseYear
g_1	'Zenimax Online Studios'	'Elder Scrolls Online'	'PC/Mac'	2014
g_2	'Zenimax Online Studios'	'Elder Scrolls Online'	'Xbox'	2015
g_3	'Arkane Studios'	'Dishonored 2'	'PC/Mac'	2016
g_4	'Lucas Arts'	'Monkey Island'	'DOS'	1990

Assume a game has one developer.

Give a DML statement that leads to an **insert**, **update**, and **delete anomaly** in instance gameRelease

Duplicated data and update anomaly

Exercise:

gameRelease

	Developer	Game	Platform	ReleaseYear
g_1	'Zenimax Online Studios'	'Elder Scrolls Online'	'PC/Mac'	2014
g_2	'Zenimax Online Studios'	'Elder Scrolls Online'	'Xbox'	2015
g_3	'Arkane Studios'	'Dishonored 2'	'PC/Mac'	2016
g_4	'Lucas Arts'	'Monkey Island'	'DOS'	1990

Better schema:

gameRelease(Game, Platform, Year)

gameDeveloper(Game, Developer)

Reducing the NULL values in tuples

- *Relations* must be designed to **minimize NULL**s
- *Attributes* with *null* can be located in separated *relations*
- Example:

  If only 15 percent of employees have individual offices, there is little justification for including an attribute Office_number in the EMPLOYEE relation;

  Better, a relation EMP_OFFICES(Essn, Office_number) can be created.

Reducing the NULL values in tuples

Minimize NULLs: Why?

- NULLs don’t have certain meaning - could be exists but unknown
- Aren’t used in joins
- Aren’t counted in aggregate functions
- Waste space

Reducing the NULL values in tuples

Minimize NULLs

| Ssn | Ename | Bdate | Addr | Dmanaged | Dno |

- What's wrong with the above schema?

  Dmanaged has many **nulls** because most employees aren’t managers

- What can be improved from the above schema?

Reducing the NULL values in tuples

Minimize NULLs

STUDENT(SID, Name, Phone, Email, SocietyName, MembershipNo)

- What's wrong with the above schema?
- What can be improved from the above schema?

Reducing the NULL values in tuples

Minimize NULLs

STUDENT(SID, Name, Phone, Email, SocietyName, MembershipNo)

↓

STUDENT(SID, Name, Phone, Email)

MEMBERSHIP(SID, SocietyName, MembershipNo).

Disallowing spurious tuples

- Splitting a relation into two relations, with the intention of joining them together if needed. However, applying a natural join generate many **more tuples** and we cannot recover the original table.
- Relations must be designed to satisfy *lossless join condition* (that is, not including ***spurious tuples***) from natural join among relations
- Lossless join: A join from a decomposition which faithfully represents the original information

Disallowing spurious tuples

Example:

BOOK

ISBN	Price	Page_count
001-987-760-9	25	800
001-354-921-1	22	200
002-678-745-2	25	810

Try to decompose this relation into two subrelations

Disallowing spurious tuples

Example:

BOOK

ISBN	Price	Page_count
001-987-760-9	25	800
001-354-921-1	22	200
002-678-745-2	25	810

decomposed into:

BOOK_PRICE

ISBN	Price
001-987-760-9	25
001-354-921-1	22
002-678-745-2	25

PRICE_PAGE

Page_count	Price
800	25
200	22
810	25

Try performing a natural join to BOOK_PRICE and PRICE_PAGE

Disallowing spurious tuples

Example:

BOOK

ISBN	Price	Page_count
001-987-760-9	25	800
001-354-921-1	22	200
002-678-745-2	25	810

decomposed into:

BOOK_PRICE

ISBN	Price
001-987-760-9	25
001-354-921-1	22
002-678-745-2	25

PRICE_PAGE

Page_count	Price
800	25
200	22
810	25

Natural joined into:

ISBN	Price	Page_count
001-987-760-9	25	800
001-987-760-9	25	810
001-354-921-1	22	200
002-678-745-2	25	800
002-678-745-2	25	810

See something odd? ***Spurious tuples!***

Disallowing spurious tuples

Example:

R(A, B, C, D)

A	B	C	D
a1	b1	c1	d1
a2	b2	c2	d1
a3	b1	c1	d2
a4	b2	c2	d3

decomposed into:

R1

B	C	D
b1	c1	d1
b2	c2	d1
b1	c1	d2
b2	c2	d3

R2

A	D
a1	d1
a2	d1
a3	d2
a4	d3

Disallowing spurious tuples

Example:

R(A, B, C, D)

A	B	C	D
a1	b1	c1	d1
a2	b2	c2	d1
a3	b1	c1	d2
a4	b2	c2	d3

R1

B	C	D
b1	c1	d1
b2	c2	d1
b1	c1	d2
b2	c2	d3

R2

A	D
a1	d1
a2	d1
a3	d2
a4	d3

R1 and R2 Join

A	B	C	D
a1	b1	c1	d1
a2	b1	c1	d1
a1	b2	c2	d1
a2	b2	c2	d1
a3	b1	c1	d2
a4	b2	c2	d3

Spurious tuples!

Disallowing spurious tuples

Example:

R = (A, B, C)

A	B	C
a1	b2	c1
a2	b2	c1
a3	b4	c2

S = (D, C)

D	C
d1	c1
d2	c2
d4	c2
d5	c3

join between R and S

RS(A, B, C, D)

A	B	C	D
a1	b2	c1	d1
a2	b2	c1	d1
a3	b4	c2	d2
a3	b4	c2	d4

(d5, c3) is lost after join

Summary

The problems which can be detected are as follows:

- **Anomalies** that cause **redundant** work to be done during insertion into and modification of a relation, and that may cause accidental loss of information during a deletion from a relation
- **Waste of storage** space due to **NULL**s and the difficulty of performing selections, aggregation operations, and joins due to NULL values
- Generation of invalid and **spurious data** during joins on base relations with matched attributes that may not represent a proper (foreign key, primary key) relationship

Design Guidelines for Relational DBs