ISTM 6212 - Week 4
Intro Relational Databases & SQL

Daniel Chudnov, Ali Obaidi

2018

Agenda

- Introduction to Databases
- Relational Model
- Basic SQL

this slides borrow liberally
from
Database System Concepts
6th ed.
Silbershatz, Korth, and Sudarshan

highly recommended

see www.db-book.com for more
(today: chapters 1-3)

Database and Database Management System (DBMS)

- A **database** is a collection of related data with some inherent meaning
- It should be designed, build and populated for a specific purpose and for preconceived application
- **DBMS** is a general software program that facilitates the process of defining, constructing, manipulating and sharing databases

Database Applications

- Banking: Customer Information, Accounts, Loans,..
- Online shopping: Customers, Products, purchases, ..
- Course registration: Student Info, Registration, Grades,..
- Manufacturing: Production, Inventory, Supply Chain,..
- Airline reservation: Reservations, Schedules, ..
- Human resources: Employee records, Salaries,..

Example: GWEB Information System

- GWEB database for maintaining information concerning student records, registration, faculty, employee information, personal information, ..
  - Defining: define data elements, data types, constraints, ..
  - Constructing: store data in appropriate files
  - Manipulating: query and update data
  - Sharing: multiple use of the database

Run course registrations on CSV

- What might go wrong?

Course ID	Section ID	Semester	Year	Building	Room	Time
BIO-101	1	Summer	2009	Painter	514	B
BIO-107	1	Summer	2009	Painter	503	A
BIO-107	2	Summer	2009	Painter	503	B
BIO-204	1	Summer	2009	Painter	200	C
CS-190	1	Summer	2009	Packard	101	H
CS-190	2	Summer	2009	Packard	101	G
CS-319	1	Summer	2009	Packard	201	C
CS-347	1	Summer	2009	Taylor	100	C
EE-222	2	Summer	2009	Watson	220	A
EE-240	1	Summer	2009	Watson	106	H

Student ID	Student Name	Dept Name	Course ID	Section ID
15151	Mozart	Biology	BIO-101	1
15151	Mozart	Biology	BIO-107	2
15151	Mozart	Biology	BIO-204	1
76543	Singh	Comp. Sci.	CS-190	2
76543	Singh	Comp. Sci.	CS-319	1
76543	Singh	Comp. Science	CS-347	1
98345	Kim	Elec. Eng.	EE-222	2
98345	Kim	Elec. Eng.	EE-240	1
98345	Kim	Comp. Sci.	CS-347	2

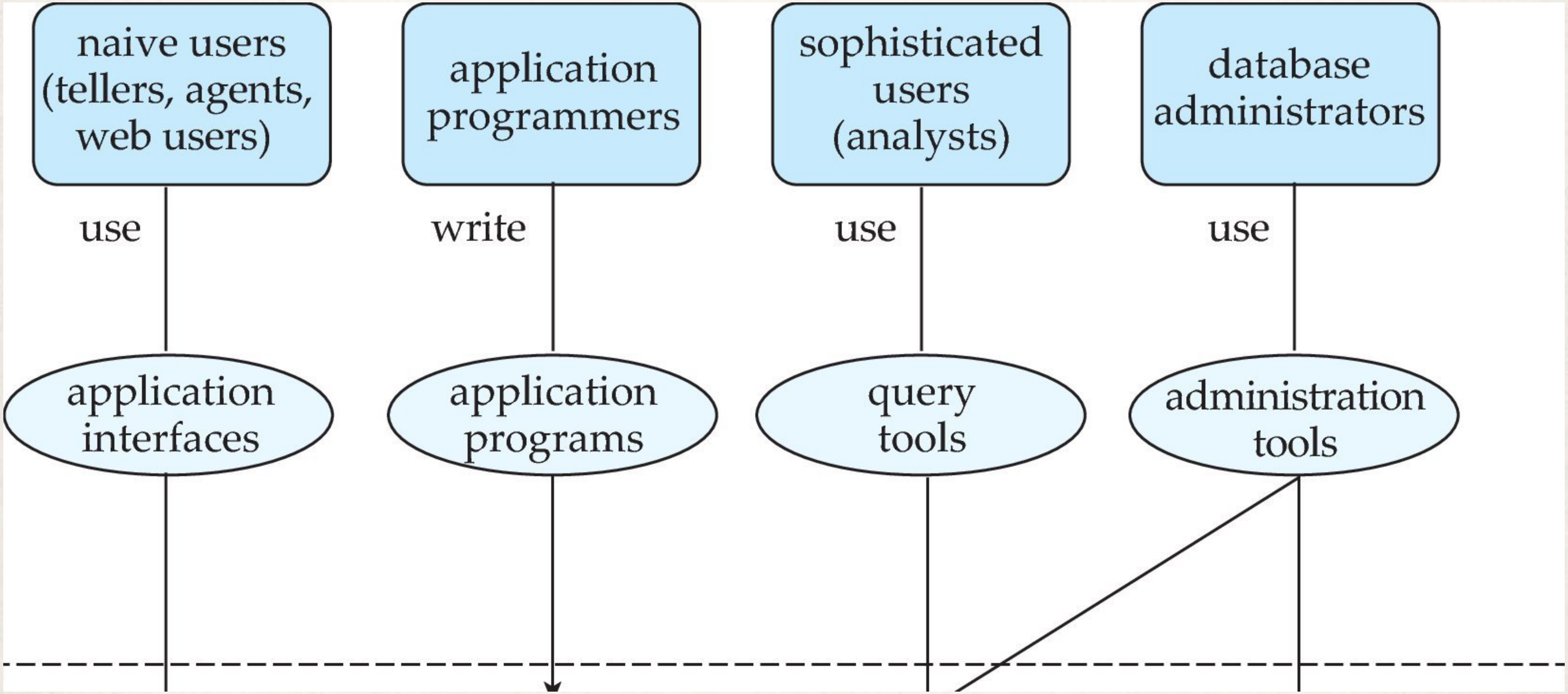
Advantages of Using DBMS

- ***Flexibility in changing structures and data requirements:*** need to add columns, remove data and change labels
- ***Providing multiple user Interfaces:*** Wait in line to update the CSV file, or risk overwriting each other
- ***Removing Data Redundancy:*** Have to enter the same data repeatedly, and could make mistakes
- ***Enforcing Integrity Constraints:*** Could register for nonexistent courses
- ***Storage and Efficient Query Processing:*** Difficult to search for information, could be slow with large number of registration records
- ***Potential for enforcing standards***
- ***Restricting Unauthorized Access:*** Non-students can register too

Databases offer:

- **Data models** (no need to enter the same data repeatedly)
- **Data Integrity** according to data models & constraints (no invalid course ID)
- **Concurrency** & Isolation (multiple users can register at the same time)
- **Transaction** - failures handled consistently & gracefully
- **Access controls** to support **security** policies (only authenticated & authorized users can access)
- Optimized data storage and retrieval (**query**) of large amount of data

Database Actors



Database

Data models

- **Data model** - a framework (collection of concepts) for describing the structure of database: data types, relationships, semantics and constraints

Categories of Data Model

- Conceptual Data Models (CDM)
  - First step in creating clear and accurate high level visual representation of the business
- Logical Data Models (LDM)
  - Extension to the CDM and represents the business requirements
- Physical Data Models (PDM)
  - describes the physical implementation of database

Levels of abstraction

- A major purpose of DBMS is to provide users with an **abstract view** of the data.
- **The internal or Physical level**
  - Describes the physical storage structure of the database. Uses a physical data model
- **The Logical level**
  - Describes the structure of the whole database for a community of users.
  - It concentrates on describing entities, data types, relationships, user operations, and constraints.
- **The external or view level**
  - Includes a number of user views describing the part of the database that a particular user group is interested in.

Relational data model

- Relational model uses a collection of **tables** to represent both **data** and the **relationships** among the data
- A **relational database system** is database management system (DBMS) based on the relational model
    - MySQL, PostgreSQL, SQLite, Oracle, SQL Server, DB2
- Not all databases are relational
    - MongoDB, Cassandra, Redis

Relational data model terms

- A **relation** *r* is a table; an **attribute** *a* is a column; a **tuple** *t* is a row;
- A relation is a ***set*** of n-tuples (*a1*, *a2*, …, *an*)
- **Domain** *D* of a attribute *a*: a set of permitted values for this attribute
  - A domain should be **atomic** - values of the domain are considered to be indivisible.
    - e.g.: "202-707-5000, 202-707-6000"
  - The **null** value means the value is *unknown* or *does not exist*.

Relation schema

- A **relation schema** is the definition of a relation.
    - **R** is a relation
    - **A1, A2,…An** are attributes
    - Relational schema: **R (A1, A2,…An)**
    - Relations are Unordered — Order of Tuples are irrelevant

e.g.:

department (*dept_name*, *building*, *budget*)

instructor (*ID*, *name*, *dept_name*, *salary*)

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

Multiple named columns
Multiple Rows (Tuples)

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Figure 1.2 A sample relational database.

Super, Candidate and Primary Keys

- A **superkey** is a set of one or more attributes whose values can uniquely identify tuples in the relation.
    - e.g. {*ID*} and {*ID, name*} are both superkeys of relation *instructor*
- A superkey is a **candidate key** if none of its proper subset is a superkey
    - e.g. {*ID*} is a candidate key of relation *instructor*
- A **primary key** is a principal candidate key chosen by designer

Foreign key

- A relation *r1* may include among its attributes the primary key of another relation *r2*. This attribute is called a **foreign key** from *r1*, referencing *r2*.

- *r1* is **referencing relation**, *r2* is **referenced relation**

department (***<u>dept_name</u>***, *building*, *budget*)

instructor (*<u>ID</u>*, *name*, ***dept_name***, *salary*)

Referential integrity constraint

- **Foreign key constraint** is a type of referential integrity constraint
- A **referential integrity constraint** requires that the values appearing in specified attributes of any tuple in the referencing relation also appear in specified attributes of at least one tuple in the referenced relation. — *Database System Concepts*

"Can't reference something that doesn't exist"

ID	name	dept_name	salary
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	building	budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Figure 1.2 A sample relational database.

Can we add an instructor from "English" department? How?

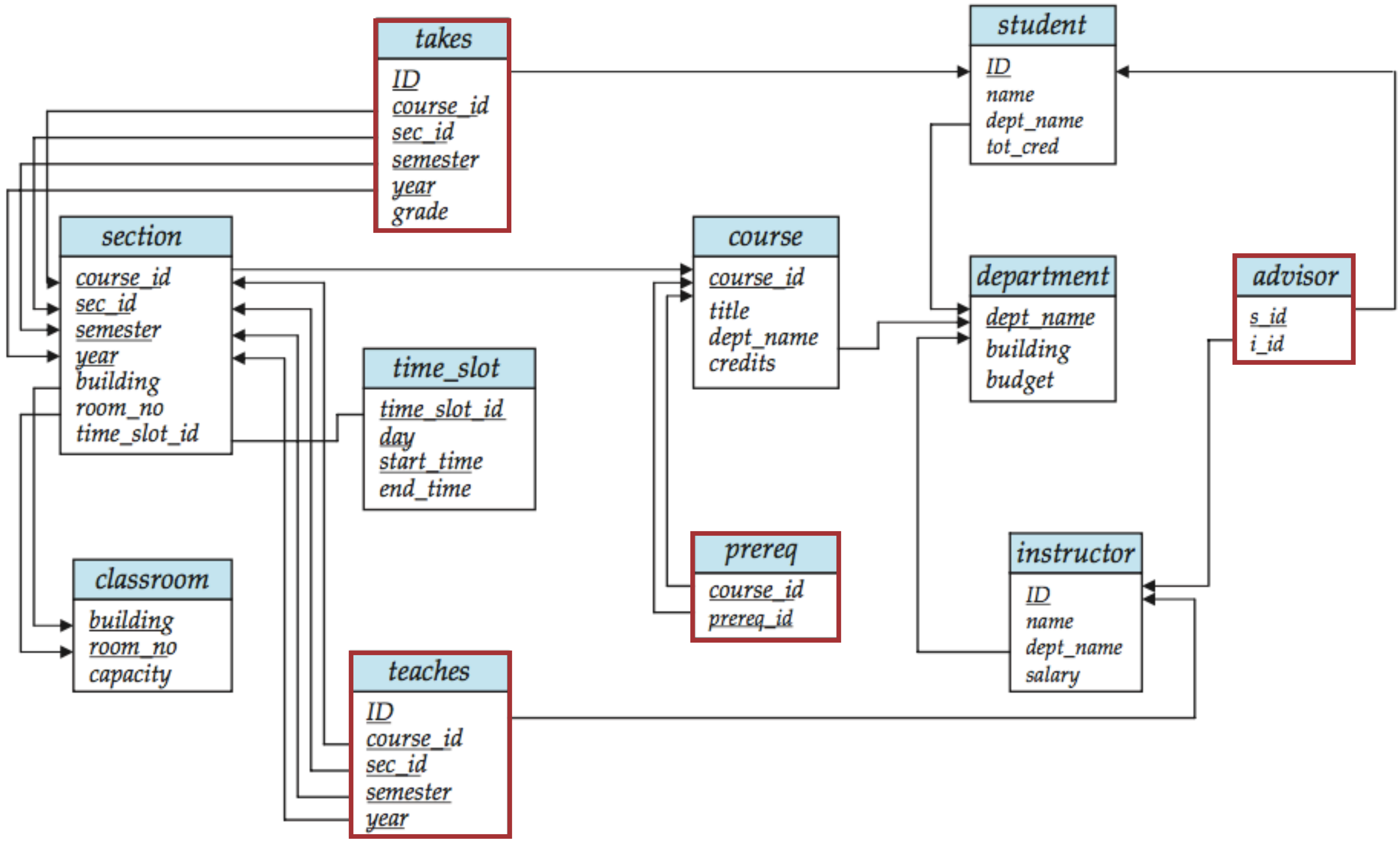


Figure 2.8 Schema diagram for the university database.

DBMS languages

- **Data-definition language** (DDL) defines the database schema

```
create table department
    (dept_name    char (20),
     building     char (15),
     budget       numeric (12,2));
```

- **Data-manipulation language** (DML) expresses database queries and updates

```
select instructor.name
from instructor
where instructor.dept_name = 'History';
```

- **Data Control Language (DCL)** grants or revokes user access to the database

```
Grant Select On Instructor
To user1 with grant option;
```

Structured Query Language (SQL)

- A combination of DDL, DML and DCL as well as statements for constraints specifications and schema evolution

Relational algebra

- Created by Edgar F. Codd at IBM in 1970
- Theoretical foundation for relational databases/SQL, as well as libraries like dplyr for R, pandas for Python
- Defines a set of operations on relations
    - Take one or two relations as input
    - Result is a new relation
    - Operations can be combined (think command line pipes)

Selection: select tuples

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

relation *r*

A	B	C	D
α	α	1	7
β	β	23	10

select tuples with A=B and D>5

$$\sigma_{A=B,D>5}(r)$$

Projection: select attributes

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

relation r

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

select (project) A and C

$$\Pi_{A,C}(r)$$

Join: Cartesian product

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

relations *r, s*

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

$r \times s$

Union

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
α	2
β	1
β	3

relations *r, s*

$r \cup s$

Set difference

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
β	1

relations *r, s*

$$r - s$$

Set intersection

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

relations *r, s*

A	B
α	2

$r \cap s$

Summary of Relational Algebra Operators

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{\text{salary}>=85000}(\textit{instructor})$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID,\ salary}(\textit{instructor})$
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
⋈ (Natural Join)	$\textit{instructor} \bowtie \textit{department}$
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
× (Cartesian Product)	$\textit{instructor} \times \textit{department}$
	Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes)
∪ (Union)	$\Pi_{name}(\textit{instructor}) \cup \Pi_{name}(\textit{student})$
	Output the union of tuples from the two input relations.

break

Basic SQL

SQL - Structured Query Language

- Grew out of IBM's Sequel language in 1970s
- International standard since 1986
- 1992 version was first widely implemented
- Basic features the same, others vary per database

What does SQL do?

- Data Definition Language (DDL) - create and manage schema for each relation, including attribute domains, key integrity constraints, indexes, security, storage

- Data Manipulation Language (DML) - query, insert, delete, update

Today DML, next week DDL

Basic SQL SELECT structure

- Typical form: SELECT $A_1, A_2, \ldots, A_n$
  FROM $r_1, r_2, \ldots, r_m$
  WHERE P;

- A_i - attribute, r_i - relation, P - predicate

- Result of SQL query is another relation

- Note: SQL is case-insensitive; multiple lines are allowed

SQL projection: SELECT

- An asterisk in the select clause denotes "all attributes"
- e.g.: SELECT *
  FROM *instructor*;
- SELECT clause could list specific **attributes** to show in query result; this is the **projection** (Π) operation
- e.g.: SELECT *dept_name*
  FROM *instructor*;
-

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 2.1 The *instructor* relation.

dept_name
Comp. Sci.
Finance
Music
Physics
History
Physics
Comp. Sci.
History
Finance
Biology
Comp. Sci.
Elec. Eng.

Figure 3.3 Result of "**select** *dept_name* **from** *instructor*".

Note the duplicates

SELECT clause: DISTINCT

- Unlike relational algebra, SQL allows duplicate rows
- DISTINCT eliminates **duplicates** in query results
- e.g.: SELECT DISTINCT *dept_name*
  FROM *instructor*;
- You can use distinct on multiple attributes

Concatenation and Literals

- Use the concatenation operator || to concatenates columns or character strings to columns.
- Literal is a string, number or a date that is included in SELECT statement that are not a column name or column alias.
- e.g.: SELECT *name* || ' – ' || *dept_name* *AS "Name and Department"*
- FROM *instructor*;

SELECT clause (cont.)

- The SELECT clause can contain arithmetic expressions or functions
- e.g.: SELECT *name, salary* / 12
  FROM *instructor;*

  SELECT *name, ROUND(salary* / 12, 2)
  FROM *instructor;*

Arithmetic Operators

- There are 4 basic arithmetic operators
  - '+' Addition
  - '-' Subtraction
  - '*' Multiplication
  - '/' Division
- Arithmetic operators can be used in the “SELECT” and “WHERE” clause
- Operator Precedence is : * and / take over the precedence of + and –
- Precedence is override by the use of parenthesis

Rename Operation: AS

- AS renames **attributes** in query results
- Double quotes is used when the name contains spaces
- e.g.: SELECT *name AS "Full Name",*
  salary / 12 AS *monthly_salary*
  FROM *instructor;*

Null Value

- It is possible for rows to have a null value, denoted by null, for some of their attributes
- null signifies an unknown value or that a value does not exist
- The result of any arithmetic expression involving null is null
- Aggregate functions simply ignore null values
- Comparisons with null values return null

SQL selection: WHERE

- "WHERE" clause specifies **conditions** result must satisfy
- It restrict the number of rows returned
- This is the **selection** (σ) operation
- e.g.:

```
SELECT name
FROM instructor
WHERE dept_name = 'History';
```

SQL selection: Boolean Ops

- AND, OR, NOT
- e.g.:

```
SELECT name
FROM instructor
WHERE dept_name = 'History'
        AND salary > 100000;
```

Boolean Ops: more examples

- e.g.:

SELECT *name*
FROM *instructor*
WHERE *salary* < 30000
    OR *salary* > 200000;

SELECT *name*
FROM *instructor*
WHERE NOT *dept_name* = 'History';

Comparison Operators

- In “WHERE” clause you may use comparison operators to compare one expression to another value or expression (and Visa Versa)
- Operators are
- **'='** (equal) **'>'** (greater than)
- **'<'** (less than) **'>='** (greater or equal)
- **'<='** (less or equal) **'<>'** (not equal)
- **NOT** **AND** **OR** (logical Operators)
- Precedence (Arithmetic, Concatenation, comparison, logical)

Truth Table

A	B	A AND B	A OR B	NOT A
F	T	F	T	T
T	T	T	T	F
F	NULL	F	NULL	T
T	NULL	NULL	T	F
F	F	F	F	T
NULL	NULL	NULL	NULL	NULL

Nulls in SQL Queries

- SQL allows queries to check if a value is NULL (missing or undefined or not applicable)
- SQL uses IS or IS NOT to compare NULLs because it considers each NULL value distinct from other NULL values, so equality comparison is not appropriate
- e.g.:

```
SELECT *
FROM Visited
Where Dated IS NULL;
```

String wildcards

- Wildcards match strings in attribute values
- '%' - match any substring, '_' - match any single character
- e.g.:
  SELECT *name*
  FROM *instructor*
  WHERE *name* LIKE ‘K%’;

ORDER BY

- Specify the ordering of result relation by attributes
- May be ASCending (default) or DESCending order
- e.g.:

  SELECT *name, dept_name*
  FROM *instructor*
  ORDER BY *dept_name DESC, name;*

go to notebook

Survey schema diagram