Course Introduction

CENG315 INFORMATION MANAGEMENT

General Information

- Weekly Course Hours:

  Mondays 9:45 – 12:00 Lecture in D0 or MS Teams

- Instructor: Belgin Ergenç Bostanoğlu
- Email: belginergenc@iyte.edu.tr
- Teams Code: **ghf0bfv**

General Information

- Teaching Assistants:
  - Büşra Güvenoğlu
  - Leyla Tekin
- Emails: busraguvenoglu@iyte.edu.tr, leylatekin@iyte.edu.tr
- Teaching Assistants’ Office Hours:
  - Will be announced on Teams

General Information: Textbooks

- A. Silberschatz, HF. Korth, S. Sudarshan, Database System Concepts, 7th Ed., McGraw-Hill, 2019.
- Edward Sciore, Database Design and Implementation, Wiley, 2nd Ed. 2020.
- Jeffrey D. Ullman and Jennifer Widom, A First Course in Database Systems, 3rd Ed., 2007.
- C.J. Date, An Introduction to Database Systems, 8th Ed., 2003.

General Information: Grading Policy

- Midterm: 35%
- Final: 40%
- Project: 25%
  - There can be minimum 5, maximum 6 students in each group.
  - Different project for each group
    - Step 1: Project Group & Title
    - Step 2: Project Design Report
    - Step 3: Final Project Design Report
    - Step 4: Implementation of the Project
    - Step 5: Presentation

Weekly Schedule

CENG315 Information Management (Fall 2022 – 2023)					
Week	Who?	Date	Subject	Book Chapter	Term Project
1	B.E.B (F2F)	3/10	Course Introduction Introduction, Relational Model, Querying	Book 2, Ch. 1, 2	
2	B.E.B (F2F)	10/10	Relational Design	Book 3, Ch. 6	
3	B.E.B.(Online)	17/10	Relational Algebra	Book 2, Ch. 4	
4	B.E.B.(Online)	24/10	Relational Algebra	Book 2, Ch. 4	Step 1: Project Group & Title
5	Assistants (F2F)	31/10	SQL	Lecture Notes	
6	Assistants (F2F)	7/11	SQL	Lecture Notes	Step 2: Project Design Report
7	B.E.B (Online)	14/11	Integrity, Security	Book 2, Ch. 5	
8	B.E.B. & Asisstants (F2F)	21/11	Midterm		Feedback on Project Design Reports
9	B.E.B (F2F)	28/11	Functional Dependencies	Book 4, Ch. 3	
10	B.E.B ((Online)	5/12	Views, Indexes	Book 2, Ch. 6	
11	B.E.B.(Online)	12/12	Transactions, Recovery	Book 1, Ch. 15	
12	B.E.B (Online)	19/12	Concurrency Control	Book 1, Ch. 16 Book 3, Ch. 18	
13	B.E.B.((Online)	26/12	Query Optimization	Book 2, Ch. 24	Step 3 & 4: Final Project Design Report & Implementation of the Project
14	B.E.B. & Assistants (F2F)	02/01	Project Presentations		Step 5: Presentation
15	B.E.B. & Asistants (F2F)	09/01	Final		

Books:

Book 1: C.J. Date, An Introduction to Database Systems, 8th Ed., 2003.

Book 2: Edward Sciore, Database Design and Implementation, Wiley, 2020.

Book 3: A. Silberschatz, HF. Korth, S. Sudarshan, Database System Concepts, 7th Ed., McGraw-Hill, 2019.

Book 4: Jeffrey D. Ullman and Jennifer Widom, A First Course in Database Systems, 3rd Ed., 2008.

Grading: Midterm: 35%, Project: 25%, Final 40%

Course Assistants: Büşra Güvenoğlu, Leyla Tekin

Introduction to Database Systems & Relational Databases

Databases and Database Systems

- Database
  - A collection of data stored in a computer
- The data in a database is typically organized into records.
  - Employee records
  - Medical records
  - Sales records
- Figure 1-1 depicts a database that holds information about students in a university and the courses they have taken.

STUDENT	SId	SName	GradYear	MajorId
	1	joe	2004	10
	2	amy	2004	20
	3	max	2005	10
	4	sue	2005	20
	5	bob	2003	30
	6	kim	2001	20
	7	art	2004	30
	8	pat	2001	20
	9	lee	2004	10

DEPT	DId	DName
	10	compsci
	20	math
	30	drama

COURSE	CId	Title	DeptId
	12	db systems	10
	22	compilers	10
	32	calculus	20
	42	algebra	20
	52	acting	30
	62	elocution	30

SECTION	SectId	CourseId	Prof	YearOffered
	13	12	turing	2004
	23	12	turing	2005
	33	32	newton	2000
	43	32	einstein	2001
	53	62	brando	2001

ENROLL	EId	StudentId	SectionId	Grade
	14	1	13	A
	24	1	43	C
	34	2	43	B+
	44	4	33	B
	54	4	53	A
	64	6	53	A

Figure 1-1
Some records for a university database

University Database

- Five types of records
- A conceptual picture of some records
  - Does not indicate:
    - How the records are stored
    - How they are accessed
- Database system
  - Software that manages the records in a database

Requirements of a Database System

- Must be persistent
- Can be very large
- Get shared
- Must be kept accurate
- Must be usable

Record Storage

- Databases must be persistent.
- Storing database records in text files:
  - The simplest and most straightforward approach
  - One file per record type
  - Each record could be a line of text, with its values separated by tabs

1 $\boxed{\text{TAB}}$ j o e $\boxed{\text{TAB}}$ 2 0 0 4 $\boxed{\text{TAB}}$ 1 0 $\boxed{\text{RET}}$ 2 $\boxed{\text{TAB}}$ a m y $\boxed{\text{TAB}}$ 2 0 0 4 $\boxed{\text{TAB}}$ 2 0 $\boxed{\text{RET}}$ ...

Figure 1-2
Implementing the STUDENT records in a text file

Record Storage: Text Files

- Advantages:
  - The database system has to do very little
  - A user could examine and modify the files with a text editor
- Disadvantages:
  - Updating the file takes to much time
  - Searching the file takes too much time

Data Models and Schemas

- Two different ways of expressing the university database:
  - As several collections of records, as in Figure 1-1
  - As several files where each record is stored in a representation as in Figure 1-2
- Each of these ways can be specified as a ***schema*** in a ***data model***.
- A ***data model*** is a framework for describing the structure of databases.
- A ***schema*** is the structure of a particular database.

Data Models and Schemas (Cont.)

- Data models:
  - Relational data model
  - File-system data model
- Schemas:
  - Expressed in terms of tables of records
  - Expressed in terms of files of records

Data Models and Schemas (Cont.)

File-system data model

- Student records are stored in the text file “student.txt”.
- There is one record per line.
- Each record contains four values, separated by tabs, denoting the student ID, name, graduation year, and major ID.
- Programs that read (and write to) the file are responsible for understanding and decoding this representation.

Relational data model

- The records are stored in a table named STUDENT.
- Each record contains four fields: an integer *SId*, a string *SName*, and integers *GradYear* and *MajorId*.
- Users access a table in terms of its records and fields: They can insert new records into a table, and retrieve, delete, or modify all records satisfying a specified predicate.

```
public static List<String> getStudents1997() {
      List<String> result = new ArrayList<String>();
      FileReader rdr = new FileReader("students.txt");
      BufferedReader br = new BufferedReader(rdr);
      String line = br.readLine();
      while (line != null) {
            String[] vals = line.split("\t");
            String gradyear = vals[2];
            if (gradyear.equals("1997"))
                  result.add(vals[1]);
            line = br.readLine();
      }
      return result;
}
```

(a) Using a file system model

```
select SName from STUDENT where GradYear = 1997
```

(b) Using the relational model

Figure 1-3
Two ways to retrieve the name of students graduating in 1997

Data Models and Schemas (Cont.)

- Most of the Java code deals with decoding the file:
  - Reading each record from the file
  - Splitting it into an array of values to be examined (Figure 1-3 (a))
- The SQL code only specifies the values to be extracted from the table
  - It says nothing about how to retrieve them (Figure 1-3 (b))

Data Models and Schemas (Cont.)

- These two models are clearly at different levels of abstraction.
- The relational model is called a ***conceptual model***
  - Its schemas are specified and manipulated without any knowledge of how it is to be implemented
- The file-system is called a ***physical model***
  - Its schemas are specified and manipulated in terms of a specific implementation

Data Models and Schemas (Cont.)

- A ***conceptual schema*** describes what the data “is”.
- A ***physical schema*** describes how the database is implemented.
- Conceptual schemas are much easier to understand and manipulate than physical schemas
  - They omit all the implementation details

Physical Data Independence

- A conceptual schema is certainly nicer to use than a physical schema.
- Operations on a conceptual schema get implemented by the database system.
- The ***database catalog*** contains descriptions of the physical and conceptual schemas.
- Given an SQL query, the database system uses its catalog to generate equivalent file-based code. This translation process enables physical data independence.
- A database system supports ***physical data independence*** if users do not need to interact with the database system at the physical level.

Benefits of Physical Data Independence

- Ease of use
  - Result from not needing to be concerned with implementation details
- Query optimization
  - Automatic optimization
- Isolation from changes to the physical schema
  - Physical implementation does not affect the user

Logical Data Independence

- Suppose that the dean's office constantly deals with student transcripts.
- They would really appreciate being able to query the following two tables:
  - STUDENT_INFO (SId, SName, GPA, NumCoursesPassed, NumCoursesFailed)
  - STUDENT_COURSES (SId, YearOffered, CourseTitle, Prof, Grade)
- The set of tables personalized for a particular user is called the user's ***external schema***.
- A database system supports ***logical data independence*** if users can be given their own external schema.

The Three Schema Levels

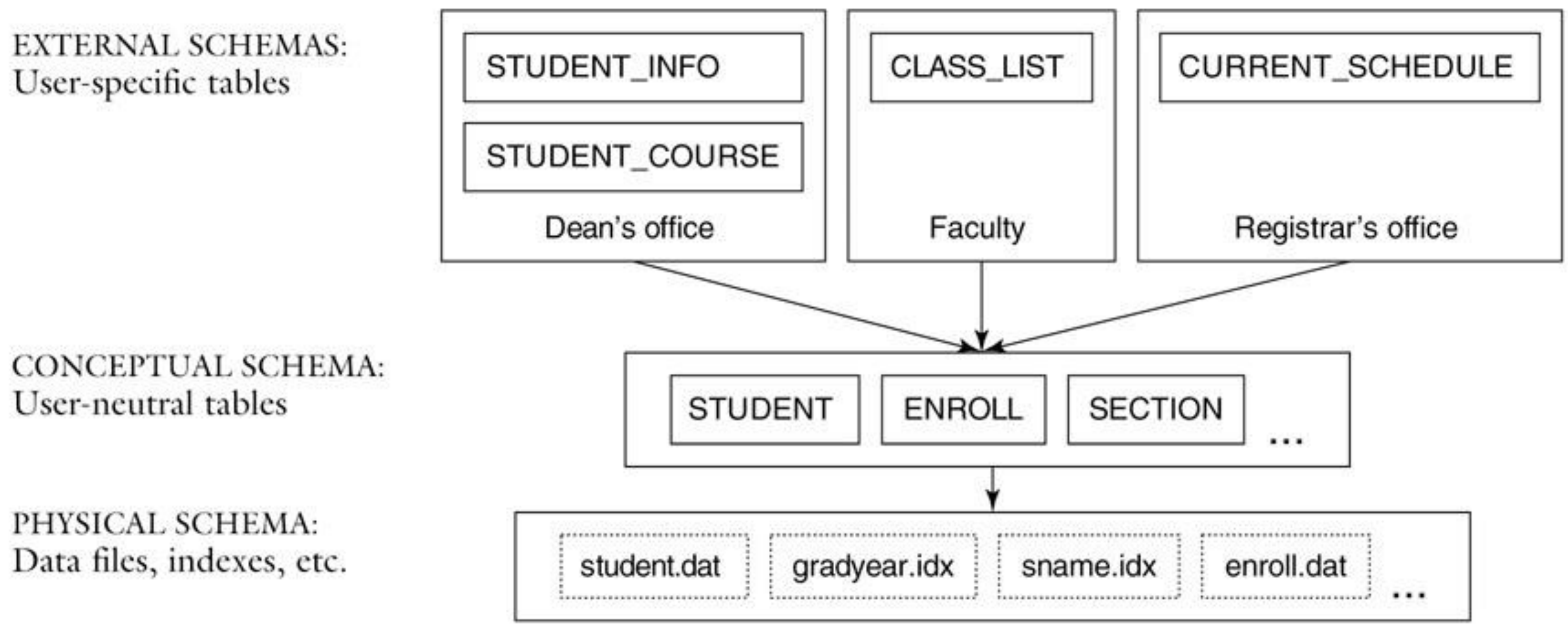

Figure 1-4
The three schema levels

Benefits of Logical Data Independence

- Each user gets a customized external schema.
  - Users see the information they need in the form that they need it, and they don't see the information they don't need.
- The user is isolated from changes to the conceptual schema.
- Users receive privileges on their schema only, which provides better security.
  - External schemas can be used to hide sensitive data from unauthorized users.

Relational Databases

Tables

- The data in a relational database system is organized into ***tables***.
- Each table contains zero or more ***records*** (the ***rows*** of the table) and one or more ***fields*** (the ***columns*** of the table).
- Each record has a value for each field.
- Each field has a specific ***type***.
- Commercial database systems support many types, including various numeric, string, date/time types.

Tables (Cont.)

- Often, when discussing a database, it is convenient to ignore the type information; in such cases, we can write the schema by simply listing the field names for each table.

```
STUDENT(SId, SName, GradYear, MajorId)
DEPT(DId, DName)
COURSE(CId, Title, DeptId)
SECTION(SectId, CourseId, Prof, YearOffered)
ENROLL(EId, StudentId, SectionId, Grade)
```

Figure 2-1
The schema of the university database

Null Values

- A ***null*** value denotes a value that does not exist or is unknown.
- Null values occur for two reasons:
  - Data collection may be incomplete.
  - Data may arrive late.

Superkeys and Keys

- A user must reference a record by specifying field values.
  - Example: “I want the record for student named Joe who graduated in 1977”.
- However, not all field values are guaranteed to uniquely identify a record.
- A unique identifier is called a ***superkey***.
- A ***superkey*** of a table is a field (or fields) whose values uniquely identify the table’s records.
- Note that adding a field to a superkey always produces another superkey.
- A ***key*** is a superkey having property that no subset of its fields is a superkey.

Superkeys and Keys (Cont.)

- STUDENT (SId, SName, GradYear, MajorId)
  - Every student has a different ID so SId is a superkey of STUDENT.
  - SId is a key.
  - {SId, GradYear} is a superkey of STUDENT.
- SECTION (SectId, CourseId, Prof, YearOffered)
  - If a professor teaches at most one section a year, then {Prof, YearOffered} is a key.
  - If a course can have at most one section per year, then {CourseId, YearOffered} is a key.

Primary Key

- Although a table can have several keys, one key is chosen to be the ***primary key***.
- Records are referenced by their primary key.
  - Each primary key should be as natural and easy to understand.
- ID numbers are often used as primary keys because they are simple and intuitive.
- Primary key fields must never be null.

Foreign Key

- The information in a database is split among its tables.
- However, these tables are not isolated from each other.
- A **foreign key** is a field (or fields) of one table which corresponds to the primary key of another table.

MajorId	in STUDENT	is a foreign key of DEPT;
DeptId	in COURSE	is a foreign key of DEPT;
CourseId	in SECTION	is a foreign key of COURSE;
StudentId	in ENROLL	is a foreign key of STUDENT;
SectionId	in ENROLL	is a foreign key of SECTION.

Figure 2-2
Foreign keys for the university database

Foreign Keys and Referential Integrity

- The specification of a foreign key asserts ***referential integrity***, which requires each non-null foreign key value to be the key value of some record.

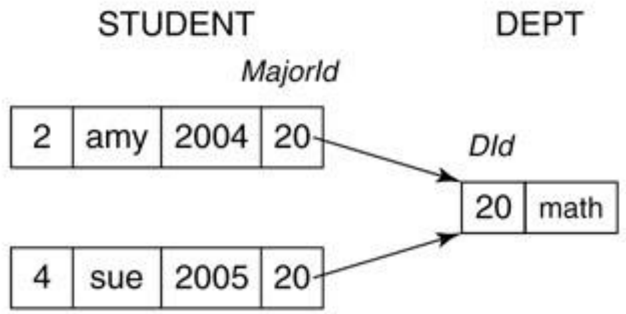

Figure 2-3
An example of referential integrity

Constraints

- A ***constraint*** describes the allowable states that the tables in the database may be in.
- There are four important kinds of a constraint:
  - ***Null value constraints*** specify that a particular field must not contain nulls.
  - ***Key constraints specify*** that two records cannot have the same values for the key's fields.
  - ***Referential integrity constraints*** specify that a foreign key value of one record must be the key value of another record.
  - ***Integrity constraints***

Integrity Constraints

- An ***integrity constraint*** encodes “business rules” about the organization.
- Integrity constraints have two purposes:
  - They can detect bad data entry.
  - They can enforce the “rules” of the organization.
- An integrity constraint may apply to
  - an individual record, “a student’s graduation year is at least 1863”
  - a table, “a professor teaches at most two sections per year”
  - or database, “a student cannot take a course more than once”

Specifying Tables in SQL

```
create table STUDENT (
      SId int not null,
      SName varchar(10) not null,
      MajorId int,
      GradYear int,

      primary key (SId),
      foreign key (MajorId) references DEPT
            on update cascade
            on delete set null,
      check (SId > 0),
      check (GradYear >= 1863)
)
```

Figure 2-4
The SQL specification of the STUDENT table

Specifying Tables in SQL (Cont.)

- The action specified with the *on delete* and *on update* keywords can be one of the following:
  - **Cascade:** causes the same query (delete or update) to apply to each foreign key record
  - **Set null:** causes the foreign key values to be set null
  - **Set default:** causes the foreign key values to be set to their default value
  - **No action**: causes the query to be rejected if there exists an affected foreign key record

References

- Edward Sciore, Database Design and Implementation, Wiley, 2020.
  - Chapter 1 & 2