Technical University of Denmark

DTU Compute
Department of Applied Mathematics and Computer Science

The Relational Model

A Data Model for Relational Databases

©**Anne Haxthausen** and Flemming Schmidt

These slides have been prepared by Anne Haxthausen, partly reusing some slides by Flemming Schmidt, which again partly reused some slides by Silberschatz, Korth. Sudarshan, 2010.

Contents

- Basic Concepts
  - database, DBMS
  - data model
  - the Relational Model: relations/tables, relation schema, relation instance, domains, NULL values, keys, database schema diagram.
- Database Languages (based on relational model)
  - Basic concepts and SQL
- Demo Exercises
- Exercises

Databases and Data Models

- A database is a collection of structured, interrelated data.
  - Example: A university database could contain info about (1) entities like students and courses and (2) relations between the entities like which courses students take.
- A database management system (DBMS) is a collection of programs used to access and manipulate data in a database.
- Each database system has an underlying logical data model.
- A data model is a way to organise the data in a database.

  Examples of data models:
  - **relational model:** the most commonly used, the one we use in 02170
  - object-based data models
  - network model (old)
  - hierarchical model (old)
  - …

The Relational Model

- Founders of the Relational Model turned the IT world upside down
  - They started making a great effort *to use mathematics as a foundation* for the Relational Model. Then they used best practice for implementation.
- The Relational Model is simple and based on set theory
  - All the data is logically structured in *relations (tables)* representing *entities* and their *relationships*.
  - A database is considered as *a set of relations*.
  - Relational Algebra offers simple operations on data.
- Databases based on the relational model are called *relational databases*.

Relations - Mathematical Definition

Mathematical Definition

- Let $D_1, D_2, \ldots, D_n$ be sets of values.
- The Cartesian Product $D_1 \times D_2 \times \ldots \times D_n$ is the set of Tuples $(a_1, a_2, \ldots, a_n)$ where each value $a_i \in D_i$.
- A relation r over $D_1 \times D_2 \times \ldots \times D_n$ is a subset of $D_1 \times D_2 \times \ldots \times D_n$ i.e. a relation is a set of tuples $(a_1, a_2, \ldots, a_n)$, where each $a_i \in D_i$.
  Each tuple *represents* a relationship between the elements $a_1, a_2, \ldots, a_n$
- A relation can be represented by a table with a row for each tuple.

Example:

The relation { (apple, 5), (banana, 4) } can be represented by

apple	5
banana	4

Relational Database Terminology

- A Relation is seen as a Table
  - with a name
  - with named Attributes (columns) of data
  - the rows are called Tuples
  - each Tuple (row) represents an entity or a relationship

- Example:

  Relation Instructor has the Attributes: InstID, InstName, DeptName and Salary, and the first Tuple represents instructor Srinivasan, who works in the Computing Science Department, has an Instructor ID 10101, and a Salary of 65000.00.

Instructor

InstID	InstName	DeptName	Salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

4	Attributes/columns
12	Tuples/rows

Terminology: Tables and Relations

- Three ways of saying almost the same thing
  - A Table of Cells stored in Rows and Columns
  - A Relation of Atoms stored in Tuples and Attributes
  - A Table of Data Elements stored in Rows and Attributes

Data in a …	with …	and …
Table	Rows	Columns
Relation	Tuples	Attributes

Relation Schema and Relation Instance

- A Relation Schema $R(A_1, ..., A_n)$
  - Defines the overall design of a relation.
  - Consists of
    - the name R of the relation, e.g. Instructor
    - the attribute names, $A_1, ..., A_n$, e.g. InstID
    - a domain D_i for each attribute A_i
    - attributes constituting primary key are underlined
  - **Example:**
    Instructor(<u>InstID</u>, InstName, DeptName, Salary)
  - Is changed infrequently, if at all!
- A Relation Instance r of a relation R
  - $r \subseteq D_1 \times D_2 \times \ldots \times D_n$
  - Is the data stored in the relation at a particular moment in time.
  - Over time the relation instance may change frequently!

Instructor

InstID	InstName	DeptName	Salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

Domains

- A Domain (similar to type)
  - Is a Set of allowed Attribute Values
- Domain Examples
  - InstID: Integers from [10000, 99999]
  - DeptName: Elements from the set {Biology, Comp. Sci., Elec. Eng., Finance, History, Music, Physics}
  - Attribute Values must be atomic

Instructor

InstID	InstName	DeptName	Salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

- All Domains have a NULL value
  - A NULL value signifies a not existing value or a unknown value
  - To insert instructor Hansen with ID 79797 into the relation Instructor without knowing his department and salary, in MySQL simply insert the tuple:
    INSERT Instructor VALUES ('79797', 'Hansen', NULL, NULL);

Keys

- Relation Schemas

  Instructor(InstID, InstName, DeptName, Salary)

  Department(DeptName, Building, Budget)

- What makes Tuples unique?
  - In Instructor each tuple has a unique InstID
  - In Department the DeptName value makes each tuple unique.

- A Key
  - Is the Attribute, or set of Attributes, that makes relation tuples unique.
  - No two tuples have the same Key
  - Sometimes it takes more than one attribute to make a Key. For example:

  Classroom(Building, Room, Capacity)

Instructor

InstID	InstName	DeptName	Salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

Department

DeptName	Building	Budget
Biology	Watson	90000.00
Comp. Sci.	Taylor	100000.00
Elec. Eng.	Taylor	85000.00
Finance	Painter	120000.00
History	Painter	50000.00
Music	Packard	80000.00
Physics	Watson	70000.00

Keys

Let $R(A_1, A_2, \ldots, A_n)$ be a relation schema and $K \subseteq \{A_1, A_2, \ldots, A_n\}$, e.g. $K = \{A_2, A_5\}$.

- (Super)key:
  - K is a superkey of R, if values for K are sufficient to identify a unique tuple of each permissible relation instance r of R. In other terms: no two rows have the same superkey.
  - Example: {InstID} and {InstID, InstName} are both superkeys of Instructor.
- Candidate Key:
  - A Superkey K is a candidate key if K is minimal.
  - Example: {InstID} is a candidate key for Instructor.
- Primary Key (Constraints):
  - One of the candidate keys is selected by the DB designer to be the primary key.
  - In the relational schema, attributes constituting the primary key are underlined, like in Instructor(<u>InstID</u>, InstName, DeptName, Salary).
  - This implies a constraint on the allowed relation instances:
    (1) no two rows may have the same value for the primary key,
    (2) the primary key value must not be NULL.
- Foreign Key (Constraints): see next page.

Foreign Keys, Example

Let $R(A_1, A_2, \ldots, A_n)$ be a relation schema, $K \subseteq \{A_1, A_2, \ldots, A_n\}$

- K can be specified to be a foreign key referencing another relation R', if K is a primary key of R'.
- This implies a referential integrity constraint on the allowed relation instances r of R and r' of R': for any tuple in r, there must exist a tuple in r' having the same values for K (unless r.K is NULL).

Example:

- Given Relation Schemas

  Instructor(<u>InstID</u>, InstName, DeptName, Salary)
  Department(<u>DeptName</u>, Building, Budget)
- Attribute DeptName in relation Instructor is a foreign key referencing the Department relation.
- This implies a referential integrity constraint on the allowed relation instances of Instructor and Department: for any tuple in Instructor, there must exist a tuple in Department having the same values for DeptName.
- Insert and delete violations: Deleting the last tuple of Department or inserting a tuple having DeptName *Mathematics* in Instructor would break the constraint.

Instructor

InstID	InstName	DeptName	Salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

Department

DeptName	Building	Budget
Biology	Watson	90000.00
Comp. Sci.	Taylor	100000.00
Elec. Eng.	Taylor	85000.00
Finance	Painter	120000.00
History	Painter	50000.00
Music	Packard	80000.00
Physics	Watson	70000.00

Relational Database Schemas and Diagrams

- A Database Schema consists of the relation schemas of all database tables.
- A Database Schema Diagram depicts the relation schemas + primary keys + foreign keys of all database tables.

Database Schema Diagram for a University

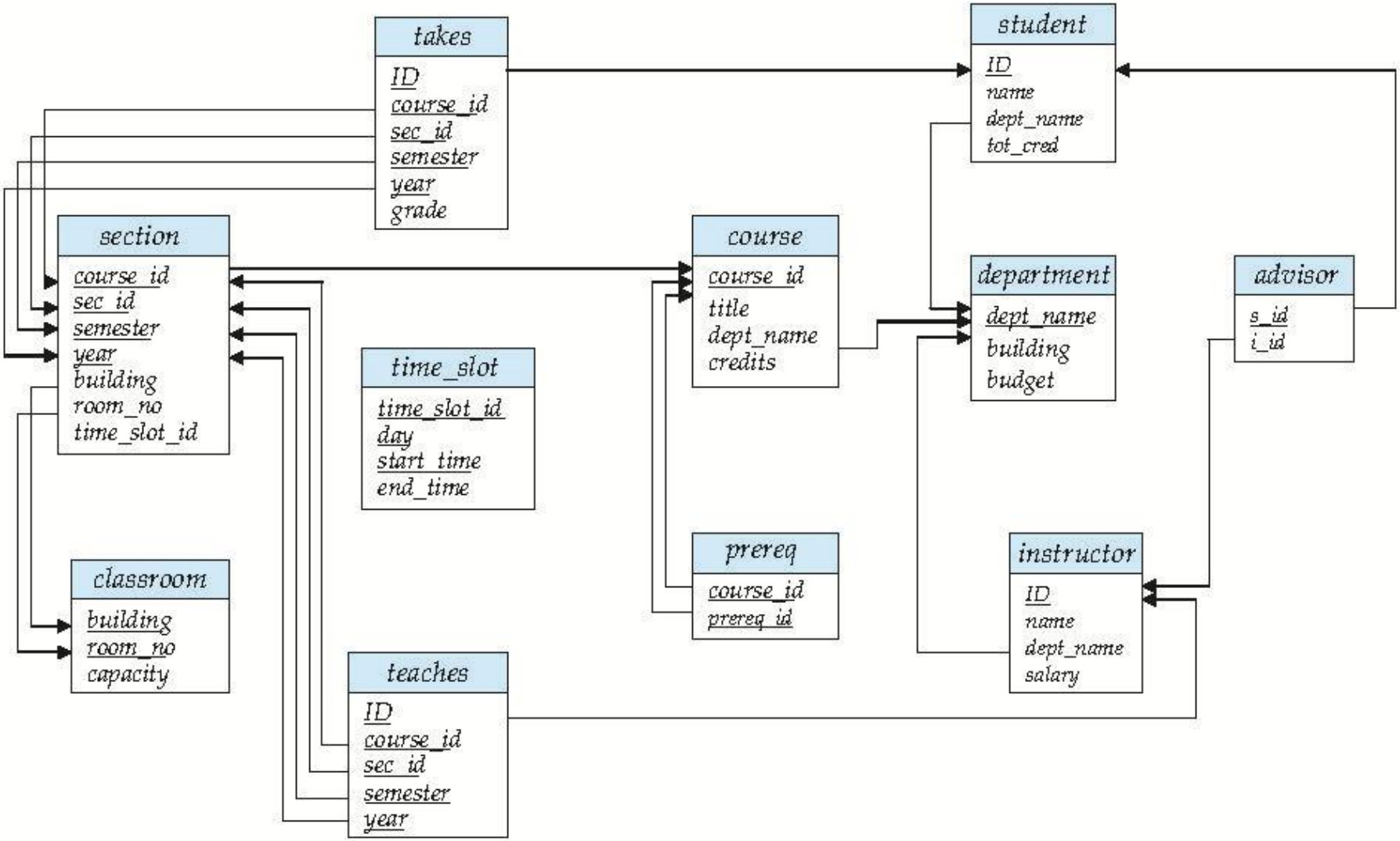


Table Names are shown in blue boxes. Primary Key Attributes are underlined.

Foreign Keys are shown with arrows.

Database Languages

- A query language is a language used to extract data from a database.
  For relational query languages, the output is a table/relation.
- SQL – Structured Query Language: most widely used query language
  - Example of a query to find physics instructors with a salary < 80000:

```
SELECT InstID, InstName FROM Instructor
   WHERE DeptName = 'Physics' AND Salary < 80000;
```

- Formal Query Languages based on Mathematics:
  - Relational Algebra with Relations as variables, for example:

$$\Pi_{\text{InstID,InstName}}\left(\sigma_{\text{DeptName}} = \text{'Physics'} \wedge \text{Salary} < 80000(\text{Instructor})\right)$$

  - Tuple Calculus with Tuple Variables (t and s), for example:
    {t | ∃ s ∈ Instructor(t[InstID] = s[InstID] ∧ t[InstName] = s[InstName]
    ∧ s[DeptName] = 'Physics' ∧ s[Salary] < 80000)}

  - Domain Calculus with Domain Variables (id, in, dn and sa):
    {< id, in > | ∃dn, sa (< id, in, dn, sa >∈Instructor
    ∧ dn = 'Physics' ∧ sa < 80000)}

SQL – Structured Query Language

- SQL is a special-purpose programming language designed for managing data stored in a relational database.
- Originally SQL was based upon *relational algebra and tuple calculus.*
- SQL was one of the first commercial languages for Edgar F. Codd's relational model, as described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks“.
- SQL quickly became the most widely used database language.
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.
- Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, though, most SQL code is not completely portable among different database systems without adjustments.

SQL Command Categories

- Data Definition Language (DDL)

  CREATE DATABASE, DROP DATABASE,
  CREATE TABLE, ALTER TABLE, DROP TABLE,
  CREATE INDEX, ALTER INDEX, DROP INDEX,
  CREATE VIEW and DROP VIEW

- Data Manipulation Language (DML)

  INSERT, UPDATE and DELETE

- Data Query Language

  SELECT (i.e. SELECT attributes FROM tables WHERE condition GROUP BY attributes)

- Data Control Language

  CREATE USER, RENAME USER, DROP USER
  ALTER PASSWORD, GRANT, REVOKE and CREATE SYNONYM

- Data Administration Commands

  START AUDIT and STOP AUDIT

- Transactional Control Commands

  SET TRANSACTION, COMMIT, ROLLBACK and SAVEPOINT

Schema & Instance in SQL Context

- Relation Schema

  Instructor(InstID, InstName, DeptName, Salary)

- Used for creating an SQL Table:

```
CREATE TABLE Instructor (
    InstID      VARCHAR(5) PRIMARY KEY,
    InstName    VARCHAR(20),
    DeptName    VARCHAR(20),
    Salary      DECIMAL(8,2));
```

Instructor

InstID	InstName	DeptName	Salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

- Relation Instance
  - Content of a relation Instructor shown in SQL:

```
SELECT InstID, InstName, DeptName, Salary FROM Instructor;
```

SQL Data Definition Language

- Data Definition Language DDL
  - Defines the Database Schema to the Data Dictionary. Example:

```
CREATE TABLE Instructor (
        InstID          VARCHAR(5),
        InstName        VARCHAR(20) NOT NULL,
        DeptName        VARCHAR(20),
        Salary          DECIMAL(8,2),
        PRIMARY KEY(InstID),
        FOREIGN KEY(DeptName) REFERENCES Department(DeptName)
        ON DELETE SET NULL);
```

SQL Data Manipulation Language

- Data Manipulation Language DML
  - Language for manipulating data in the database

- Like for Instructor(InstID, InstName, DeptName, Salary):
  - Insert a new instructor
    INSERT Instructor VALUES ('79797', 'Hansen', 'Finance', NULL);
  - Move 'Wu' to the Physics department and set salary to 95000
    UPDATE Instructor SET DeptName = 'Physics', Salary = 95000.00 WHERE InstID = 12121;
  - Wu has left, delete him from the table
    DELETE FROM Instructor WHERE InstID = 12121;

SQL Data Query Language

- Data Query Language DQL
  - Given Instructor(InstID, InstName, DeptName, Salary)
  - Find the name of the instructor with InstID equal to 22222:
    SELECT InstName FROM Instructor WHERE InstID = '22222';

InstName
Einstein

  - Find the InstID and Building of instructors in the Physics department:

```
SELECT InstID, Building FROM Instructor, Department
  WHERE Department.DeptName = 'Physics' AND
        Instructor.DeptName = Department.DeptName;
```

InstID	Building
22222	Watson
33456	Watson

Long names when ambiguity
Short: AttributeName
Long: TableName.AttributeName

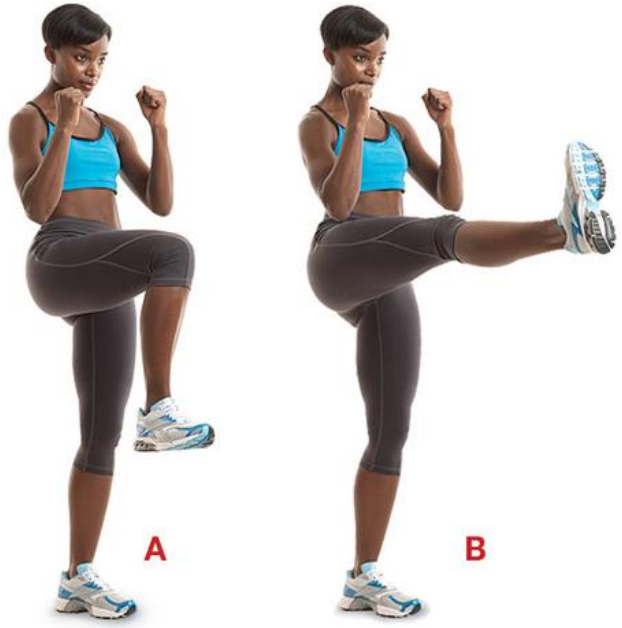
Summary

- **Basic Concepts:** database, DBMS, data model, relational model, relations/tables, relation schema, relation instance, domains, NULL values, keys, database schema diagram.
- **Language Concepts in SQL.**

Readings

- In Database Systems Concepts **edition 7**: sec. 1.1-1.5 + 2.1-2.5,2.7
- In Database Systems Concepts **edition 6**: sec. 1.1-1.6 + 2.1-2.7
- Pay special attention to the Summaries.

Demo Exercises



Demo Exercises clarify ideas and concepts from the Lecture to provide you with good Database Skills.

Discuss and do the Demo Examples.

Relational Model Terminology

1.1 Terminology

Make assumptions and describe the data below using Relational Terminology.

Student

StudID	StudName	Birth	DeptName	TotCredits
00128	Zhang	1992-04-18	Comp. Sci.	102
12345	Shankar	1995-12-06	Comp. Sci.	32
19991	Brandt	1993-05-24	History	80

Instructor

InstID	InstName	DeptName	Salary
10101	Srinivasan	Comp. Sci.	65000.00
32343	El Said	History	60000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
83821	Brandt	Comp. Sci.	92000.00

Advisor

StudID	InstID
00128	45565
12345	10101

Department

DeptName	Building	Budget
Comp. Sci.	Taylor	100000.00
History	Painter	50000.00
Physics	Watson	70000.00

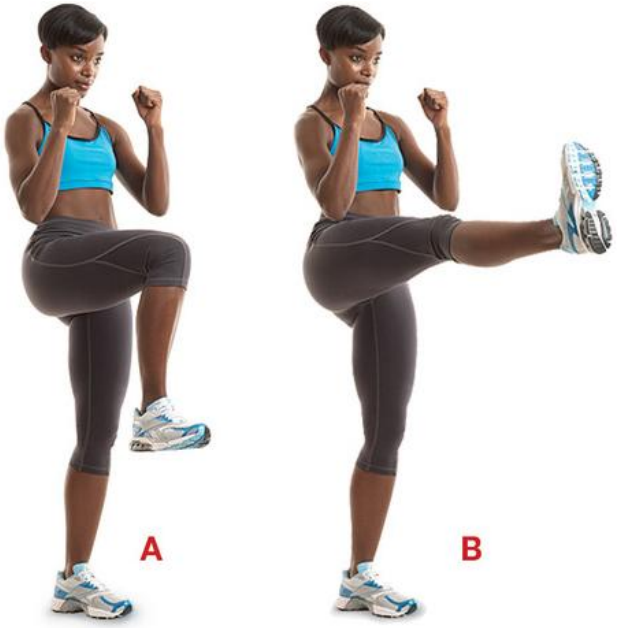
Database Scheme Diagram

1.2 Database Schema Diagram .

Draw a Database Schema Diagram showing Relations, Attributes, Primary Keys and Foreign Keys using the Relation Schemas:

```
Student(StudID, StudName, Birth,
DeptName, TotCredits)

Instructor(InstID, InstName, DeptName,
Salary)

Advisor(StudID, InstID)

Department(DeptName, Building, Budget)
```

Solutions to Demo Exercises



Relational Model Terminology

1.1 Terminology

Make assumptions and describe the data below using Relational Terminology.

Student

StudID	StudName	Birth	DeptName	TotCredits
00128	Zhang	1992-04-18	Comp. Sci.	102
12345	Shankar	1995-12-06	Comp. Sci.	32
19991	Brandt	1993-05-24	History	80

Instructor

InstID	InstName	DeptName	Salary
10101	Srinivasan	Comp. Sci.	65000.00
32343	El Said	History	60000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
83821	Brandt	Comp. Sci.	92000.00

Advisor

StudID	InstID
00128	45565
12345	10101

Department

DeptName	Building	Budget
Comp. Sci.	Taylor	100000.00
History	Painter	50000.00
Physics	Watson	70000.00

1. A Relational Database Instance!
2. With 4 Relations (tables) named Student, Instructor, Advisor and Department.
3. Relation Student has the Schema: `Student(StudID, StudName, Birth, DeptName, TotCredits)` defining the 5 Attributes (columns).
4. Relation Student has 3 Tuples (rows).
5. The Student Primary Key Attribute named StudID makes all Tuples in Student unique.
6. The Student Foreign Key called DeptName references the Department Relation.
7. Domain for TotCredits is Integers.
8. The Advisor Primary Key makes all Tuples unique. It is either <StudID> alone, meaning a student has just one instructor, or <StudID, InstID>, meaning that a student can have many advisors.
9. The Relation Advisor has two Foreign Keys: StudID referencing Student and InstID referencing Instructor.

Database Scheme Diagram

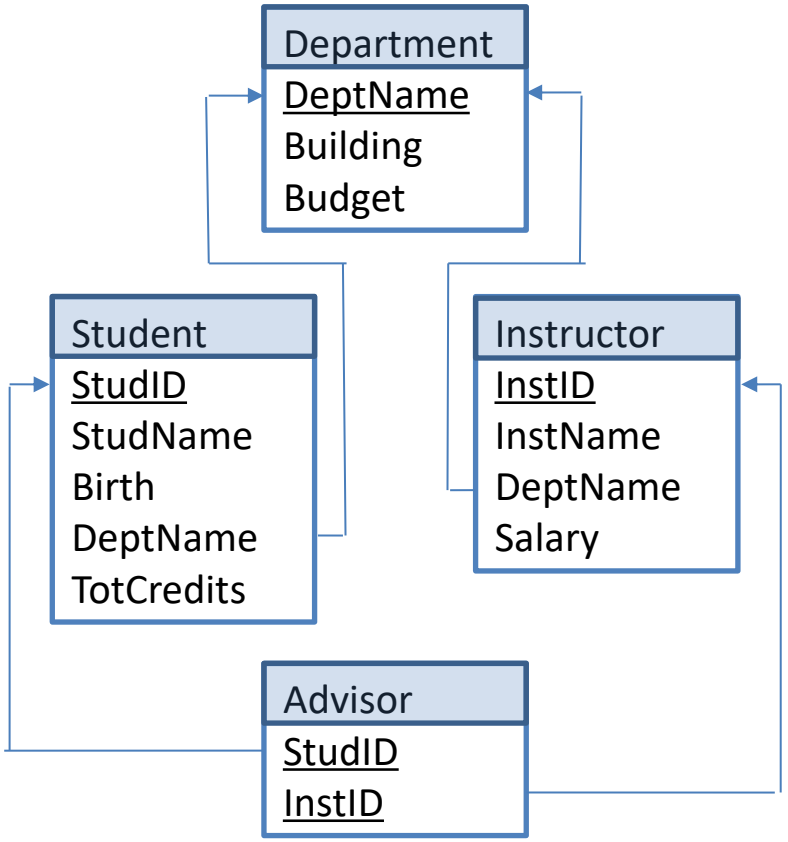
1.2 Database Schema Diagram

Draw a Database Schema Diagram showing Relations, Attributes, Primary Keys and Foreign Keys using the Relation Schemas:

```
Student(StudID, StudName, Birth,
DeptName, TotCredits)

Instructor(InstID, InstName, DeptName,
Salary)

Advisor(StudID, InstID)

Department(DeptName, Building, Budget)
```

Database Schema Diagram

Write Relation Names in blue boxes.
Write Attribute Names below the relation name with Primary Key Attributes underlined.
Depict Foreign Keys by arrows.



Exercises

Please answer all exercises to demonstrate your Database Skills.

Pencil and Paper Exercises

Solutions are available at 11:00

Primary Keys and Foreign Keys

1.4 Primary Keys

Underline potential Primary Key attributes in the Relation Schemas:

Employee (FullName, Street, City)
Works (FullName, CompanyName, Salary)
Company (CompanyName, City)

1.5 Insert and Delete Violations

Give examples of insert and delete violations of the Foreign Key constraint.

```
Instructor
InstID InstName DeptName  Salary
12121  Wu       Finance   90000
15151  Mozart   Music     40000
22222  Einstein Physics   95000

Department
DeptName  Building Budget
Finance   Painter  120000
Music     Packard   80000
Physics   Watson    70000
```

Foreign Keys and Schema Diagrams

1.6 Foreign Keys

Banking Database Relation Schemas:

Branch(BName, BCity, Assets)
Customer(CName, CStreet, CCity)
Loan(LNumber, BName, Amount)
Borrower(CName, LNumber)
Account(ANumber, BName, Balance)
Depositor(CName, ANumber)

Check the table below and write the Foreign Keys and Referenced Relations.

Relation	Primary Keys	Foreign Keys	Referencing
Branch	BName		
Customer	CName		
Loan	LNumber		
Borrower	CName LNumber		
Account	ANumber		
Depositor	CName ANumber		

1.7 Database Schema Diagram

Draw a Database Schema Diagram for the Banking Database.