
Lecture 2: Relational Model

Dragan Mirkovic
Department of Computer Science
University of Houston

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Announcements

- The course TA is Yonhong Yan (yanyh@cs.uh.edu)
- Office hours will be 9 – 10:30 TTH
- Lab will probably start on 2/1/2005
 - Accounts for you will be created on the departmental Oracle server
 - You can also download your own copy of Oracle and install it on your own computer (see the info on the course web page)
 - I will make the Oracle 9i CDs available.
- Quiz #1 on 2/1/2005
 - Ch. 3.1 – 3.4 and 3.6

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Introduction

- Proposed by Codd in 1970
- Most widely used model.
 - Vendors: IBM, Informix, Microsoft, Oracle, Sybase
 - Very simple and elegant
- Legacy systems in older models
 - E.g., IBM's IMS
- Recent competitor: object-oriented model
ObjectStore, Versant, Ontos
 - A synthesis emerging: *object-relational model*
 - Informix Universal Server, UniSQL, O2, Oracle, DB2

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Definitions

- **Relational database:** a set of **relations** with distinct relation names
- **Relation:** set of **tuples** or **records**. Made up of 2 parts:
 - **Instance** : a *table*, with rows and columns.
 - #Rows = *cardinality*,
 - #fields = *degree / arity*.
 - **Schema** : specifies name of relation, plus name and type of each column (or field or attribute).
- Relation as a set of rows or tuples (i.e., all rows are distinct).
- **Relational database schema:** a collection of schemas for the relations in the database

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Schema Example

`Students(sid: string, name: string, login: string, age: integer, gpa: real).`

- Specifies the **name** of each field and its **domain**
- The field named *sid* has a domain named *string* = set of all character arrays (strings)
- Age is an integer, gpa real, etc.
- **Domain constraints** specified in schema must be satisfied by all **instances**.
- Domain of a field is the **type** of that field

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Instance Example

- Instance of Students Relation

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- Cardinality = 3, degree = 5, all rows distinct
- No two rows are identical:
 - Requirement of the relational model
 - Not always true in practice
- The order of the fields may or may not matter

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Formal definition

Let

$R(f_1:D_1, f_2:D_2, \dots, f_n:D_n)$

be relation schema, and for each $f_i, i=1, \dots, n$, let

Dom_i be a set of values associated with the domain named D_i .

An instance of R that satisfies the domain constraints in the schema is a set of tuples with n fields

$\{ \langle d_1, \dots, d_n \rangle : d_1 \in Dom_1, \dots, d_n \in Dom_n \}$

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Relational Query Languages

- A major strength of the relational model: supports simple, powerful *querying* of data.
- Queries can be written intuitively, and the DBMS is responsible for efficient evaluation.
 - The key: precise semantics for relational queries.
 - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.
- SQL (Structured Query Language)
 - Standard language for relational DBMS

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

SQL Query Language

- Developed by IBM (system R) in the 1970s
 - Need for a standard since it is used by many vendors
 - Standards:
 - SQL-86 developed by ANSI (SQL1)
 - SQL-89 (minor revision)
 - SQL-92 (major revision) (ANSI and ISO, SQL2)
 - SQL-99 (major extensions, current standard)
 - We will use SQL-99 (SQL3. There is also SQL-2003.)
 - SQL Implementations may also have their own proprietary extensions
-

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Using SQL

- Uses the word table to denote a relation
 - A subset of SQL that supports creation, deletion, and modification of tables = DDL (Data Definition Language)
 - Definition of new domains (types)
 - Equivalent to type specifications in programming languages
 - Built-in types (integer, string, ...)
-

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Using SQL ...

- **CREATE TABLE** statement
 - Defines a new table
- Example: Students relation:

```
CREATE TABLE Students ( sid   CHAR(20) ,
                        name CHAR(30) ,
                        login CHAR(20) ,
                        age  INTEGER,
                        gpa   REAL)
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Using SQL ...

- Tuples are inserted using the **INSERT** command:
- Tuples are deleted using the **DELETE** command:

```
INSERT
INTO Students (sid, name, login, age, gpa)
VALUES ('53688', 'Smith', 'smith@ee', 18, 3.2)
```

- Delete all tuples with the name = 'Smith'

```
DELETE
FROM Students S
WHERE S.name = 'Smith'
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Using SQL ...

- To modify the existing values use:

```
UPDATE Students S
SET S.age = S.age+1, S.gpa = S.gpa-1
WHERE S.sid = 53688
```

- **WHERE** determines which rows are modified
- **SET** determines how these rows are modified
- Example:

```
UPDATE Students S
SET S.gpa = S.gpa-1
WHERE S.gpa >= 3.3
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

SQL Query Language

- Relational database query = question about the data
- Answer = new relation containing the result
- Example:
 - Find all students younger than 18 enrolled in a specific course

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

SQL Query Language

- Example:
 - To find all 18 year old students, we can write:

```
SELECT *  
FROM Students S  
WHERE S.age=18
```

sid	name	login	age	gpa
53666	Jones	jones@cc	18	3.8
53688	Smith	smith@ee	18	3.2

- To find just names and logins, replace the first line:

```
SELECT S.name, S.login
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Querying Multiple Relations

- What does the following query compute?

```
SELECT S.name, E.cid  
FROM Students S, Enrolled E  
WHERE S.sid=E.sid AND E.grade="A"
```
- Given the following instances of Students and Enrolled

sid	name	login	age	gpa
53666	Jones	jones@cc	18	3.8
53688	Smith	Smith@math	18	3.4
53650	Smith	Smith@chem	19	3.6

sid	cid	grade
53666	COSC1410	B
53666	MATH1120	C
53650	COSC3320	A
53666	CHEM2210	B

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Integrity constraints

- Restrictions on the data stored in DBMS
- **Integrity constraints (IC)**
 - condition that must be true for *any* instance of the database
 - ICs are specified when schema is defined.
 - ICs are checked when relations are modified.
 - If all ICs are satisfied in an instance of the schema = the instance is **legal**
- Only legal instances are stored!
 - Enforced by DBMS
- IC types:
 - Domain constraints
 - Other: Key constraints, Foreign key, general constraints, etc.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Primary Key Constraints

- A set of fields is a *key* for a relation if :
 - No two distinct tuples can have same values in all key fields, and
 - This is not true for any subset of the key.
- Part 2 false? A *superkey*.
- If there's >1 key for a relation, one of the keys is chosen (by DBA) to be the *primary key*.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Primary Key Constraints

- Certain minimal subset of the fields should be a unique identifier for the tuple
- A **candidate key** is a set of fields of the relation that uniquely identifies a tuple.
 - Two distinct tuples cannot have identical values in all the fields of a key
 - No subset of the key is a unique identifier for a tuple
- Example for Students relation *sid* field is a candidate key
- A relation may have several candidate keys

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Keys in SQL

- Possibly many *candidate keys* (specified using **UNIQUE**), one of which is chosen as the *primary key*.
- Examples:
 - For a given student and course, there is a single grade.
 - Students can take only one grade course, and receive a single grade for that course;
 - No two students in a course receive the same grade.
- Used carelessly, an IC can prevent the storage of database instances that arise in practice!

```
CREATE TABLE Enrolled      CREATE TABLE Enrolled
(sid CHAR(20),              (sid CHAR(20),
cid CHAR(20),              cid CHAR(20),
grade CHAR(2),             grade CHAR(2),
PRIMARY KEY(sid,cid))      PRIMARY KEY (sid),
                           UNIQUE (cid, grade))
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Foreign Keys

- **Foreign key** : Set of fields in one relation that is used to `refer` to a tuple in another relation.
 - Must correspond to primary key of the second relation.
 - Like a `logical pointer`.
 - If one relation is modified the other should be checked and possibly modified
- E.g. *sid* is a foreign key referring to Students:
 - Enrolled(*sid*: string, *cid*: string, *grade*: string)
 - If all foreign key constraints are enforced, *referential integrity* is achieved, i.e., no dangling references.
 - Can you name a data model w/o referential integrity?
 - Links in HTML!

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Foreign Keys in SQL

- Only students listed in the Students relation should be allowed to enroll for courses.

```
CREATE TABLE Enrolled
  (sid CHAR(20), cid CHAR(20), grade CHAR(2),
   PRIMARY KEY (sid,cid),
   FOREIGN KEY (sid) REFERENCES Students)
```

- *sid* is a primary key in Students relation!

sid	cid	grade
53666	COSC1410	B
53666	MATH1120	C
53650	COSC3320	A
53666	CHEM2210	B

sid	name	login	age	gpa
53666	Jones	jones@cc	18	3.8
53688	Smith	Smith@math	18	3.4
53650	Smith	Smith@chem	19	3.6

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Enforcing Referential Integrity

- Consider Students and Enrolled; *sid* in Enrolled is a foreign key that references Students.
- What should be done if an Enrolled tuple with a non-existent student id is inserted? (*Reject it!*)
- What should be done if a Students tuple is deleted?
 - Also delete all Enrolled tuples that refer to it.
 - Disallow deletion of a Students tuple that is referred to.
 - Set *sid* in Enrolled tuples that refer to it to a *default sid*.
 - (In SQL, also: Set *sid* in Enrolled tuples that refer to it to a special value *null*, denoting *'unknown'* or *'inapplicable'*.)
- Similar if primary key of Students tuple is updated.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Referential Integrity in SQL

- SQL/92 and SQL:1999 support all 4 options on deletes and updates.
 - Default is **NO ACTION** (*delete/update is rejected*)
 - **CASCADE** (also delete all tuples that refer to deleted tuple)
 - **SET NULL / SET DEFAULT** (sets foreign key value of referencing tuple)

```
CREATE TABLE Enrolled(  
    sid CHAR(20),  
    cid CHAR(20),  
    grade CHAR(2),  
    PRIMARY KEY (sid,cid),  
    FOREIGN KEY (sid)  
    REFERENCES Students  
    ON DELETE CASCADE  
    ON UPDATE SET DEFAULT )
```

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Where do ICs Come From?

- ICs are based upon the semantics of the real world enterprise that is being described in the database relations.
- We can check a database instance to see if an IC is violated, but we can NEVER infer that an IC is true by looking at an instance.
 - An IC is a statement about *all possible* instances!
 - From example, we know *name* is not a key, but the assertion that *sid* is a key is given to us.
- Key and foreign key ICs are the most common; more general ICs supported too.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

General Constraints

- In some cases it is necessary to specify more general constraints
- Example:
 - Student age > minimum age
 - DBMS rejects inserts and updates that violate the constraint
- General constraints:
 - **Table constraints:** associated with a single table
 - **Assertions:** involve several tables

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Examples

- We can specify complex constraints over a single table:

```
CREATE TABLE Drivers (did CHAR(20),
    dname CHAR(40), phone CHAR(12),
    age INTEGER, PRIMARY KEY (did),
    CHECK (age >= 16))
```

- Table constraint

- Assertions: IC over several tables

```
CREATE ASSERTION AssertName CHECK ( <Assert
    Conditions>)
```

- More in Chapter 5.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Introduction to Views

- Ch. 3.6 in text. (We will come to 3.5 latter)
- View is a table whose rows are derived from another (stored) relation
 - A *view* is just a relation, but we store a *definition*, rather than a set of tuples.

```
CREATE VIEW YoungActiveStudents (name, grade)
AS SELECT S.name, E.grade
FROM Students S, Enrolled E
WHERE S.sid = E.sid and S.age < 21
```

- Views can be dropped using the DROP VIEW command.
 - How to handle DROP TABLE if there's a view on the table?
 - DROP TABLE command has options to let the user specify this.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Views and Security

- Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s).
 - Different views for users with different privileges
 - Views can be used like any other relation in defining a query
- Example:
 - Given YoungStudents, but not Students or Enrolled, we can find students s who are enrolled, but not the *cid*'s of the courses they are enrolled in.
 - For some applications an external schema can be specified using views
 - Support for **logical data independence**

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Updates on Views

- Views behave like tables for queries
 - What about the updates?
- In SQL-92:
 - Updatable views: defined on a single base table using selection and projection only (no aggregate operations and no DISTINCT operator)
- In SQL-99:
 - Updatable views may depend on more than one table
 - A field can be updated if it is obtained from only one table and the primary key of that table is included in the fields of the view
 - Insertable-into views: views into which new rows can be inserted

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Example

- Consider the following view:

```
CREATE VIEW GoodStudents (sid, gpa)
  AS SELECT  S.sid, S.gpa
  FROM Students S
  WHERE S.gpa > 3.0
```

- Depends on the on a single table (Students)
 - Contains the primary key for that table (sid)
 - Hence, it is updatable (padded with null values)
- Changes may not be visible in the view:

```
INSERT INTO GoodStudents (sid, gpa) VALUES (51234, 2.8)
```

 - Adding **WITH CHECK OPTION** to the view definition disallows invisible insertions

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Relational model: Summary

- A tabular representation of data.
- Simple and intuitive, currently the most widely used.
- Integrity constraints can be specified by the DBA, based on application semantics.
 - DBMS checks for violations.
 - Two important ICs: primary and foreign keys
 - In addition, we *always* have domain constraints.
- Powerful and natural query languages exist.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Homework

- Page 96, 3.8, 3.9 and 3.19.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Homework

- **Exercise 3.8** Answer each of the following questions briefly. The questions are based on the following relational schema:

Emp(*eid*: integer, *ename*: string, *age*: integer, *salary*: real)

Works(*eid*: integer, *did*: integer, *pctime*: integer)

Dept(*did*: integer, *dname*: string, *budget*: real, *managerid*: integer)

1. Give an example of a foreign key constraint that involves the Dept relation. What are the options for enforcing this constraint when a user attempts to delete a Dept tuple?

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Solution

```
CREATE TABLE Works (eid INTEGER NOT NULL,  
                    did INTEGER NOT NULL,  
                    pcttime INTEGER,  
                    PRIMARY KEY (eid, did),  
                    UNIQUE (eid),  
                    FOREIGN KEY (did) REFERENCES Dept)
```

- When a user attempts to delete a Dept tuple, There are four options:
 - Also delete all Works tuples that refer to it.
 - Disallow the deletion of the Dept tuple if some Works tuple refers to it.
 - For every Works tuple that refers to it, set the *did* field to the *did* of some (existing) 'default' department.
 - For every Works tuple that refers to it, set the *did* field to *null*.