Relational Model

5 November 2012
Lecture 3

Topics for Today

- Introduction to the Relational Model
- Integrity Constraints over Relations
  - Key Constraints
  - Foreign Key Constraints
  - General Constraints
- Enforcing Integrity Constraints

- Source: Ramakrishnan and Gehrke Chapter 3

The Relational Model

- Due to Codd
- Tables are referred to as **Relations**
- Relations are built from a **Schema**
  - Schema describes what each row (**Records**) looks like
- Describes what each **field** is named and its type
  - We can define custom domains for a given built in type (later)
  - We can define custom types (next semester)

Example Schema

- Students (*sid:*string, *name*:string, *login*:string, *age:*integer, *gpa:*real)

Sid	*Name*	*Login*	*Age*	*Gpa*
53	Jones	ajones@cs	18	34.5
54	Jones	bjones@cs	30	91.3
12	Smith	smith@is	23	78.2
53	Cohen	cohen@math	19	80.2

Duplicates and Ordering

- In general, you can’t have duplicate rows
  - Keys prevent this is most cases
  - How?
- Without keys, some commercial databases allow duplicates
  - Why?

- In general, the order of records does not matter
  - Why?
  - We **can** sort, if we want
  - We **can** ask do filtering based on position (next semester)

Domains

- Simple built in domains include:
  - Integers
  - Reals
  - Characters (length) / Varchar (length)
  - Date
  - Date Time
  - Boolean
  - Text
  - Binary Large Objects (BLOB)
- Can impose custom domain constraints
- Abstractly, a schema is then:

$$\{\langle f_1 : d_1, \ldots, f_n : d_n \rangle \mid d_1 \in Dom_1, \ldots, d_n \in Dom_n\}$$

Some Relational Terms

- **Degree** or **arity** of a relation – number of fields
- **Cardinality** of a relation – number of records
- **Relational Database** – collection of relations with distinct names
- **Relational Database Schema** – collection of schemas for the relations in a relational database

Creating Relations in SQL

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

- Creates the relation we saw before, minus the key

Adding a record

```
INSERT INTO Students
    (sid, name, login, age, gpa)
    VALUES
    (53, ‘Harry’, ‘harry@ise’, 18, 94.4)
```

- Order of the entries in the first parentheses based on the order of the columns in the schema
  - You can skip it if you want
- Note that ‘ (one tick) denotes a string

Deleting Records

DELETE FROM Students WHERE Students.name = ‘Harry’

- This is an implicit query:
  - First find all the rows which match ‘Harry’
  - Then delete them

- We could also use a slight variation of the command:

DELETE FROM Students WHERE name = ‘Harry’

- Since there is no ambiguity about which table name belongs to

Updating Records

```
UPDATE Students S
    SET S.age = S.age +1, S.gpa = S.gpa -1
    WHERE S.sid = 53
```

- This uses an alias (S) for Students
- This is also an implicit query:
  - First find all rows where sid is 53
  - Then for each row:
    - Get the age value
    - Add 1 to it
    - Store the result back in the age field for that row

    - Get the gpa value
    - Subtract 1 from it
    - Store the result back in the gpa field for that row

So Far

- Introduction to the Relational Model
- Integrity Constraints over Relations
  - Key Constraints
  - Foreign Key Constraints
  - General Constraints
- Enforcing Integrity Constraints

Integrity Constraints

- Integrity is enforced by the DBMS at the *relational level*
  - Key Constraints
  - Foreign Key Constraints
  - General Constraints

- Integrity constraints (IC) are listed in the schema and define what is a **legal instance** of the relation
- The DBMS checks ICs when changes are proposed
  - DBMS may **forbid the change**
  - DBMS may **modify the propose change**

- Domain constraints are a simple kind of IC
  - A field of type Integer can’t have a Real

Key Constraints

- Key constraint: an IC which requires that a certain *minimum* number of fields must uniquely identify each record

- For all legal instances, possible keys are called **Candidate Keys**
  1. Must uniquely identify all rows
  2. Must not be a superset of any other Candidate Key

- If it breaks number 2, it's a **superkey**

Key examples

- Candidate keys:
  - Login
  - Sid
- Super keys
  - {sid, name}
  - {login, age}

Sid	Name	Login	Age	Gpa
53	Jones	ajones@cs	18	34
54	Jones	bjones@cs	30	45.5
12	Smith	smith@cs	23	40
52	Cohen	cohen@math	19	84

Keys in SQL

- DBA selects one candidate key to be the **primary key**
  - DBMS will then enforce its uniqueness
  - Can be used for indexing and optimization too (not in this course)

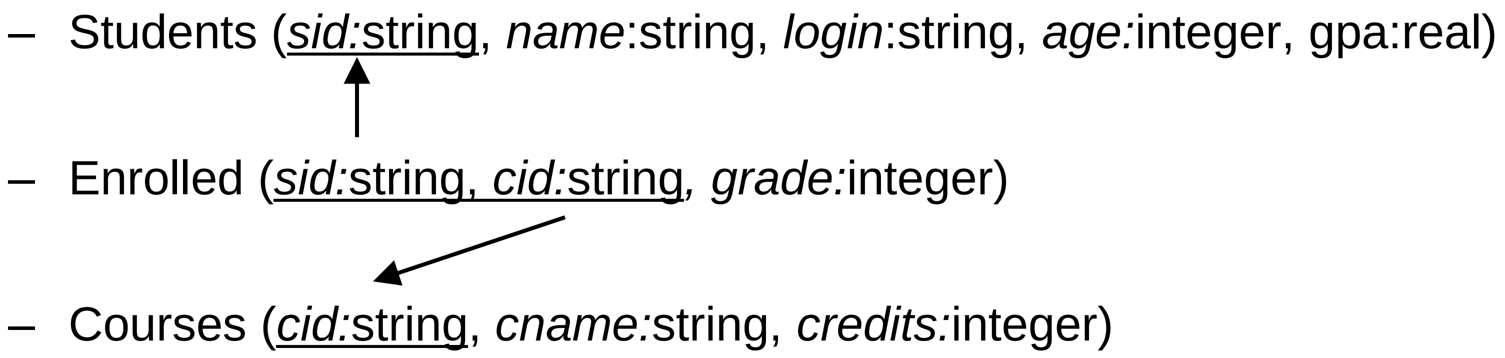
- Other candidate keys can be identified using the keyword UNIQUE in SQL
  - We can give them aliases to help identify them later

```
CREATE TABLE Students (sid CHAR(20),
    name CHAR(20),
    login CHAR(20),
    age INTEGER,
    gpa REAL,
    UNIQUE (name, age),
    CONSTRAINT StudentsKey PRIMARY KEY (sid))
```

sid CHAR(20) primary key

Foreign Key Constraints

- Relations can depend on other relations using **foreign key** relations
  - A dynamic domain restriction for a field



  - Students (*sid:*string, *name*:string, *login*:string, *age:*integer, gpa:real)
  - Enrolled (*sid:*string, *cid:*string, *grade:*integer)
  - Courses (*cid:*string, *cname:*string, *credits:*integer)

- Each value in a the pointing field must appear in the primary key field of the target
  - Or **null**

Foreign Key Enforcement

- If you try to enter an illegal value in the pointing column, the DBMS will forbid it
  - What if you delete an entry from the target relation?

- You can have a foreign key within the same table

Students (*sid:*string,
name:string,
login:string,
*age:*integer,
gpa:real,
partner:string)

All *partner* values must appear in *sid*.

How do you enter in the first row?

Foreign Key in SQL

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

- Does this allow students to received more than one grade for a given course id?
  - What would it take to allow that?

General Constraints

- SQL allows for a variety of other constraints
- **Custom domains**:
  - NOT NULL is a simple one
  - We can also write more generic ones
- **Table Constraints**: restrict values in a relation
- **Assertions**: Free form query checks
- We'll talk about these when we get to more advanced SQL
  - Not all are supported by DB vendors

So Far

- Introduction to the Relational Model
- Integrity Constraints over Relations
  - Key Constraints
  - Foreign Key Constraints
  - General Constraints
- Enforcing Integrity Constraints

Enforcing Integrity Constraints

- When adding or updating any row, DBMS must check ICs:
  - UNIQUE constraints
  - Primary Key constraints
  - NOT NULL constraints

- When deleting a row, we don't need to check UNIQUE or Primary Keys
  - But what about Foreign Keys constraints?

What to check

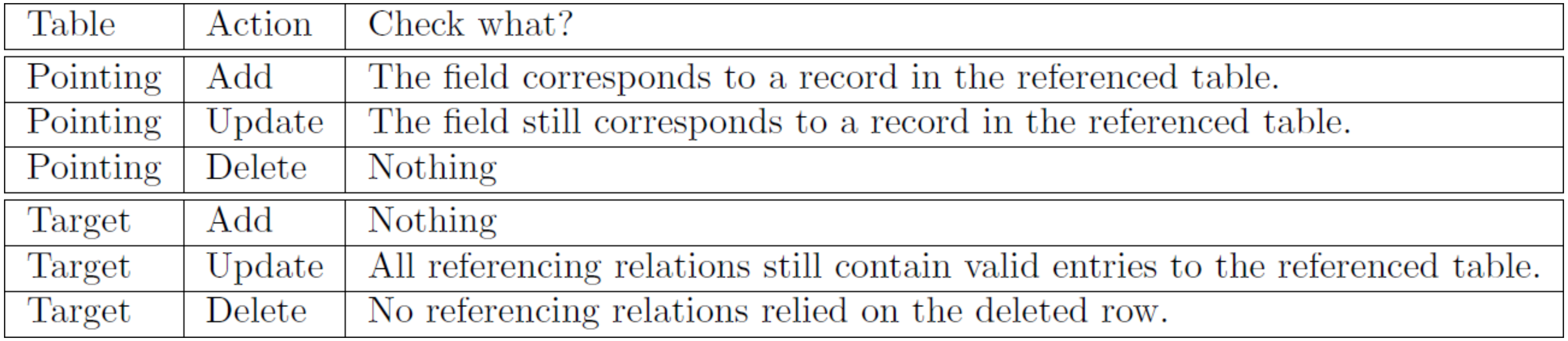


Table	Action	Check what?
Pointing	Add	The field corresponds to a record in the referenced table.
Pointing	Update	The field still corresponds to a record in the referenced table.
Pointing	Delete	Nothing
Target	Add	Nothing
Target	Update	All referencing relations still contain valid entries to the referenced table.
Target	Delete	No referencing relations relied on the deleted row.

Conclusion

- Introduction to the Relational Model
- Integrity Constraints over Relations
  - Key Constraints
  - Foreign Key Constraints
  - General Constraints
- Enforcing Integrity Constraints