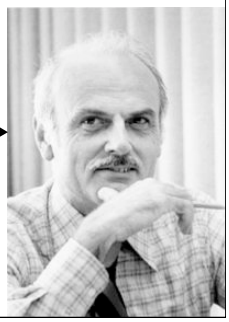
The Relational Model and SQL DDL

CISC637, Lecture #2

Ben Carterette

Relational Model

- The relational model is the most widely used in modern DBMS
  - IBM's DB2, Informix, Oracle, Sybase, Microsoft Access and SQL Server, MySQL, PostgreSQL
- Introduced by Edgar Codd in 1970
- A set-based logical model for representing data



Relations

- **Relations** are the main construct in the relational model
- Two components:
  - **Relation instance** is a table with rows and columns
  - A **relation schema** describes the columns of the table

Relation Schema

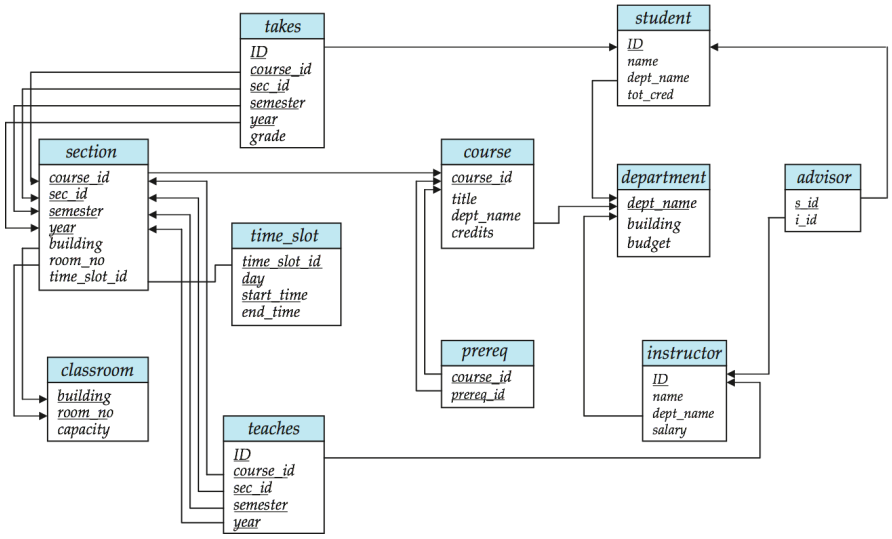
- The schema defines a relation in terms of:
  - The relation name
  - The name of each column (or field)
  - The domain of each field
- Example:
  - Student(*ID*: `integer`, *name*: `string`, *dept_name*: `string`, *tot_cred*: `integer`)

Relation Instance

- An **relation instance** is a set of records or tuples
  - Each record has the same fields as defined in the schema
  - The value in each field is in the field's domain as defined in the schema
    - Or a NULL value (allowed for some fields)
  - Every record is unique
- If any of these conditions are not true, then the relation instance is **invalid**

Relational Databases

- A **relational database** is a collection of relations with unique names
- The **relational database schema** is the collection of schema for its relations
- An **instance** of the RDB is a collection of relation instances



 9

Integrity Constraints

- **Integrity constraints** are used to ensure a database instance is valid
  - Defined on the database schema
  - Restrict the data that can be stored in the database instance
- A DBMS enforces the integrity constraints by only ever storing legal instances
- Example: *domain constraints*
  - Values in a record's field must correspond to domain as defined in the schema
  - No DBMS will store any data that violates a domain constraint (MySQL just typecasts everything)

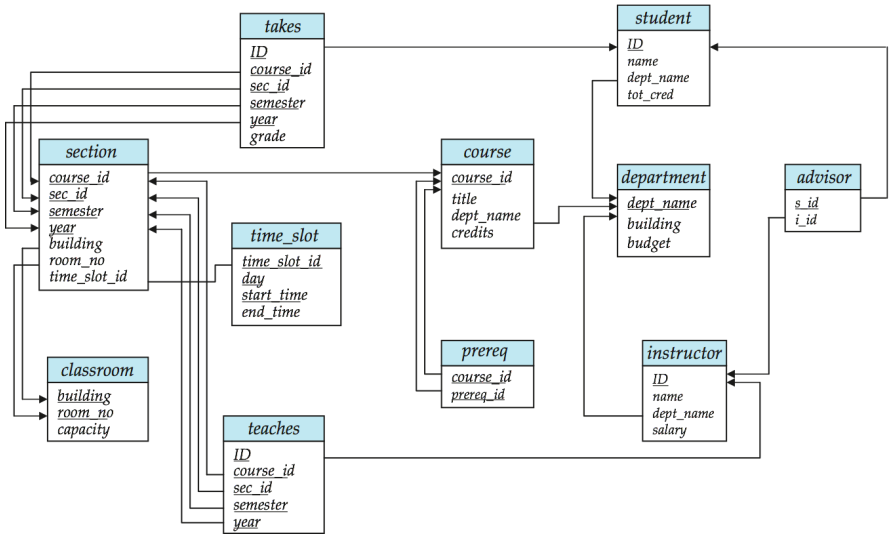
 10

Key Constraints

- A **key** is a field(s) that uniquely identifies a record
  - A **key constraint** says that every relation must have at least one key
  - It is an integrity constraint that ensures that all records are uniquely identifiable
- Three types of keys:
  - **Candidate key**:
    - Any set of fields that can uniquely identify records
    - No two distinct records can have the same values in all of a candidate key's fields,
    - but any strict subset of fields *can* have the same values in two different records.
  - **Superkey**:
    - Any set of fields that *contains* a candidate key
  - **Primary key**:
    - One of the candidate keys is chosen to be the primary key
    - Often the "smallest" candidate key, just one or a few fields

Foreign Key Constraints

- A **foreign key** is a set of fields in one relation that refers to a record in a *different* relation
  - In other words, a foreign key uniquely identifies a record in a different relation
- The purpose of foreign keys is to enforce **referential integrity**
  - Every foreign key should point to data that exists and is valid
  - No "dangling references"
- A DBMS that *enforces* foreign key constraints *guarantees* referential integrity



 13

Activity

- Same problem as Tuesday
  - Reserving a shelter at a public park
- This time, design the relational database
  - Write out tables with fields
  - Underline primary keys
  - Use arrows to indicate foreign keys
- Be sure to include *all* tables that would be necessary to use the app

 14

Shelter	Size	Price	Electricity?	Alcohol?	Accessible?
Ambridge	Large	$60	yes	no	no
Babblebrook	Small	$30	no	no	no
Beaver	Small	$30	no	no	no
Beveridge	Large	$45	no	no	no
Blackrock	Small	$30	no	no	no
Buffalo	Small	$30	no	no	no
Carnegie	Small	$30	no	yes	no
Cottage	Small	$30	no	no	no
Deer Browse #1	Small	$30	no	no	no
Deer Browse #2	Small	$30	no	no	no
Deer Browse #3	Small	$30	no	yes	no
Devil's Elbow #1	Small	$30	no	yes	no
Devil's Elbow #2	Small	$30	no	yes	no
Donnevale	Small	$30	no	no	no
Donora	Large	$60	yes	no	yes
Kolich	Small	$30	no	no	yes
Duquesne	Small	$30	no	no	no
Elwood	Small	$30	no	yes	no
Emsworth	Small	$45	yes	no	no
Erie	Small	$30	no	no	no
Etna	Small	$30	no	no	no
Fawn	Small	$30	no	no	no
Flagstaff Hill	Large	$60	yes	no	no
Forest Glen	Small	$30	no	no	no
Frazier	Small	$30	no	no	no
Garner	Large	$60	yes	no	no
Gold Star	Large	$60	yes	no	yes
Grant	Small	$45	yes	no	no
Greentree	Small	$30	no	no	no
Hampton	Small	$30	no	no	no
Harmar	XL	$70	yes	yes	yes
Harmony	Small	$30	no	no	no
Haysville	Small	$30	no	no	no

SQL: A Brief Overview

- SQL: Structured Query Language
- Developed by IBM in the 1970s (separately from Codd)
- Standard language for DBMS
  - SQL:2011 is the current standard
  - Many DBMS include their own additions and modifications to the SQL language
- SQL comprises:
  - a Data Definition Language (DDL) for defining relations
  - a Data Manipulation Language (DML) for querying relations
- Today: a brief look at the DDL

Creating Relations in SQL

- Relations are always called "tables" in SQL

```
CREATE TABLE Student ( ID        INTEGER AUTO_INCREMENT,
                       name      CHAR(30),
                       dept_name CHAR(40),
                       tot_cred  INTEGER,
                       PRIMARY KEY (ID) )
```

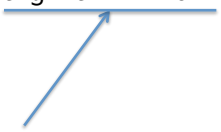

Defining Other Candidate Keys

- Use the `UNIQUE` keyword to define other sets of fields that can uniquely identify records

```
CREATE TABLE Student ( ID        INTEGER AUTO_INCREMENT,
                       ssn       INTEGER NOT NULL,
                       name      CHAR(30),
                       dept_name CHAR(40),
                       tot_cred  INTEGER,
                       PRIMARY KEY (ID),
                       UNIQUE (ssn) )
```


Defining a Foreign Key

```
CREATE TABLE Student ( ID        INTEGER AUTO_INCREMENT,
                       name      CHAR(30),
                       dept_name CHAR(40),
                       tot_cred  INTEGER,
                       PRIMARY KEY (ID),
                       FOREIGN KEY (dept_name)
                         REFERENCES Department (dept_name) )
  engine = InnoDB
```



This is required in MySQL to enforce foreign key constraints.
I am going to require that you always use it when defining tables in this class.

Defining Foreign Keys

```
CREATE TABLE Takes (
   ID INTEGER,
   course_id VARCHAR(7),
   sec_id VARCHAR(3),
   semester VARCHAR(6),
   year INTEGER,
   grade VARCHAR(3),
   PRIMARY KEY (ID, course_id, sec_id, semester, year),
   FOREIGN KEY (ID) REFERENCES Student (ID),
   FOREIGN KEY (course_id, sec_id, semester_year)
     REFERENCES Section (course_id, sec_id, semester, year) )
  engine = InnoDB
```


Adding Records to a Table

- Insert a record into an existing table

```
INSERT
INTO Student (ID, name, dept_name, tot_cred)
VALUES (53688, 'Smith', 'CIS', 18)

INSERT
INTO Takes (ID, course_id, sec_id, semester, year, grade)
VALUES (53688, 'CISC637', '010', 'Spring', '2015', 'A')
```


Display All Records in a Table

- Very simple SQL query

```
SELECT *
FROM Student
```


Deleting Records in SQL

- Delete records that satisfy some logical condition

```
DELETE
FROM Student
WHERE name = 'Smith'
```

- But what should happen if there are records in Takes referring to students named Smith?

Deleting Relations in SQL

- Deleting a table deletes schema as well as all records in that table

```
DROP TABLE Student
```


Modifying Relations in SQL

- Rename tables; add/delete/rename columns; modify field domains

```
ALTER TABLE Student ADD birthdate CHAR(10)
ALTER TABLE Student DROP tot_cred
ALTER TABLE Student MODIFY birthdate DATE
ALTER TABLE Student ADD age INT
ALTER TABLE Student RENAME RegisteredStudent
```


Modifying Records in SQL

- Change specified field values in all records that satisfy some logical condition

```
UPDATE Student
   SET tot_cred = tot_cred+3,
       birthdate = '1995-04-08'
 WHERE ID = 53688
```


Referential Integrity

- What should happen if:
  - … we delete a record in Student that is referenced by a record in Takes?
  - … we change the student ID of a Student that is referenced in Takes?
  - … we delete a record in Course that is referenced in in a Section that is referenced in Takes?
  - … we change the course ID of a Course that is referenced in a Section that is referenced in Takes?

Enforcing Referential Integrity

- What to do when deleting or updating records that are referenced in other tables?
- Four options:
  - Delete/update all records that depend on the deleted record
  - Do not allow original record to be deleted/updated (throw error)
  - Set foreign key field values to refer to a default
  - Set foreign key field values to null

Enforcing Ref. Integrity in SQL

```
CREATE TABLE Takes (
   ID INTEGER,
   course_id VARCHAR(7),
   sec_id VARCHAR(3),
   semester VARCHAR(6),
   year INTEGER,
   grade VARCHAR(3),
   PRIMARY KEY (ID, course_id, sec_id, semester, year),
   FOREIGN KEY (ID) REFERENCES Student (ID)
         ON DELETE CASCADE
         ON UPDATE CASCADE,
   FOREIGN KEY (course_id, sec_id, semester_year)
     REFERENCES Section (course_id, sec_id, semester, year)
         ON DELETE NO ACTION
         ON UPDATE NO ACTION )
 engine = InnoDB
```


Summary

- A *relational database* consists of *relations*
  - Relations are implemented as tables of records that have the same fields (but different values)
  - Each relation has a *relation instance* and a *relation schema*
- A valid relational database satisfies all integrity constraints defined on it
  - Domain constraints, key constraints, foreign key constraints
- The SQL Data Definition Language allows us to:
  - create relational databases
  - define integrity constraints
  - modify relation schema and data in relation instances

