The Relational Model: Database Definition and Integrity Constraints

Chapter 3

Relational (Data) Model

- **Schema** = a description of data in terms of a data model
- **Instance** = a table, with rows (aka tuples, records), and columns (aka fields, attributes) that match the schema
  - # of rows: cardinality
  - # of columns: degree or arity

Students

sid	name	login	age
13	Lisa	lsimp	40
41	Bart	bart	20

Courses

cid	cname	Cr.
E-484	EECS484	4
E-584	EECS584	3

Enrolled

sid	cid	Grade
41	E-484	A-
13	E-584	A+

Cardinality: 2;
Degree: 4

Relational (Data) Model

- **Schema** = a description of data in terms of a data model
- **Instance** = a table, with rows (aka tuples, records), and columns (aka fields, attributes) that match the schema
  - # of rows: cardinality
  - # of columns: degree or arity

Students

sid	name	login	age
13	Lisa	lsimp	40
41	Bart	bart	20

Cardinality: 2;
Degree: 4

Courses

cid	cname	Cr.
E-484	EECS484	4
E-584	EECS584	3

Cardinality: 2;
Degree: 3

Enrolled

sid	cid	Grade
41	E-484	A-
13	E-584	A+

Relational (Data) Model

- **Schema** = a description of data in terms of a data model
- **Instance** = a table, with rows (aka tuples, records), and columns (aka fields, attributes) that match the schema
  - # of rows: cardinality
  - # of columns: degree or arity

Students

sid	name	login	age
13	Lisa	lsimp	40
41	Bart	bart	20

Cardinality: 2;
Degree: 4

Courses

cid	cname	Cr.
E-484	EECS484	4
E-584	EECS584	3

Cardinality: 2;
Degree: 3

Enrolled

sid	cid	Grade
41	E-484	A-
13	E-584	A+

Cardinality: 2;
Degree: 3

New Scenario: Olympic Games

- Some history:
  - Inspired by the ancient Olympic Games, which were held in Olympia, **Greece** (8th century BC).

Example:
Instance of Athlete Relation

AID	Name	Country	Sport
1	Simone Biles	USA	Gymnastics
2	Gabby Thomas	USA	Track
3	Quan Hongchan	China	Diving

What is the schema?

Example:
Instance of Athlete Relation

AID	Name	Country	Sport
1	Mary Lou Retton	USA	Gymnastics
2	Jackie Joyner-Kersee	USA	Track
3	Guo Jingjing	China	Diving

What is the schema?

(aid: integer, name: `string`,
country: `string`, sport: `string`)

Relational Query Languages

- Supports simple, powerful querying of data
- Queries written declaratively
  - In contrast to procedural methods
- DBMS is responsible for efficient evaluation
  - System can optimize for efficient query execution, and still ensure that the answer does not change
- SQL is the standard database query language

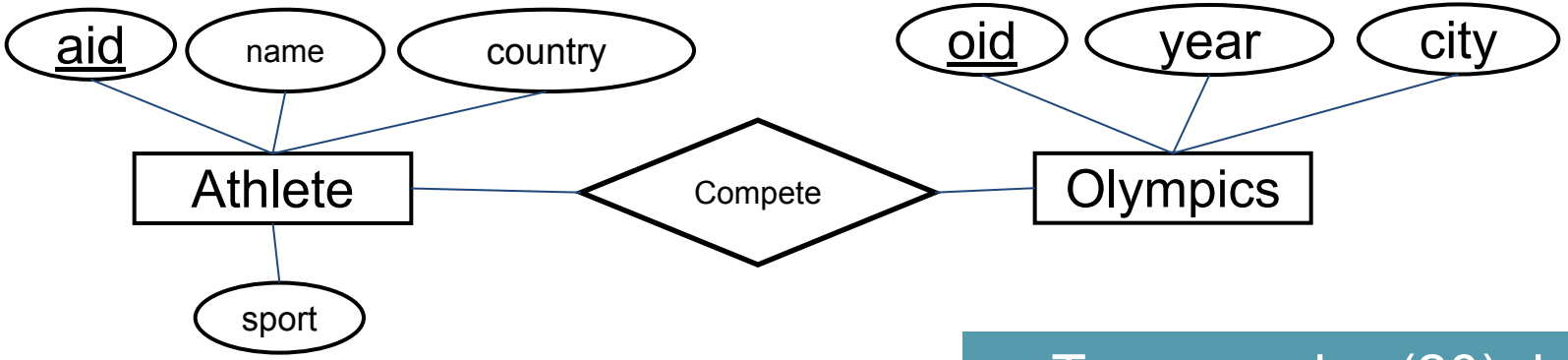
Structured Query Language (SQL)

- **CREATE a Table**
  - Integrity Constraints
- INSERT records
- SELECT records
- UPDATE records
- DELETE records
- CREATE View
- UPDATE View

Create a Table (Relation)

```
CREATE TABLE table_name (
   field1 TYPE,
   field2 TYPE,
   … … …
);
```

Try it out
in DuckDB, or in Sqlite3




```
CREATE TABLE table_name
(
   field1 TYPE,
   field2 TYPE,
   … …
);
```

- Types: char(20), integer, real, text, blob…

Creating a database, e.g.,
sqlite3 olympics.sdb
duckdb olympics.ddb

Download SQLite and DuckDB:
https://www.sqlite.org/download.html
https://duckdb.org

Creating Relations in SQL

- Create the Athlete relation
  - Domain constraint (type) enforced when tuples added or modified

```
CREATE TABLE Athlete
(aid INTEGER,
 name CHAR(30),
 country CHAR(20),
 sport CHAR(20));
```

- Create the Olympics relation

```
CREATE TABLE Olympics
(oid INTEGER,
 year INTEGER,
 city CHAR(20));
```

- Create the Compete relation

```
CREATE TABLE Compete
(aid INTEGER,
 oid INTEGER);
```

Structured Query Language (SQL)

- **CREATE a Table**
  - **Integrity Constraints**
- INSERT records
- SELECT records
- UPDATE records
- DELETE records
- CREATE View
- UPDATE View

Integrity Constraints (ICs)

- Must be true for *every* instance of the database
  - ICs are specified when schema is defined
  - ICs are checked whenever relations are modified
- A legal instance of a relation satisfies *all* specified ICs
  - DBMS must not admit illegal instances

Example Integrity Constraint

- Specify certain attributes as keys of a table

- Reference keys of entities in other tables
- No dangling references are allowed in a database when updates occur to any table

Integrity Constraint: Primary and Candidate Keys

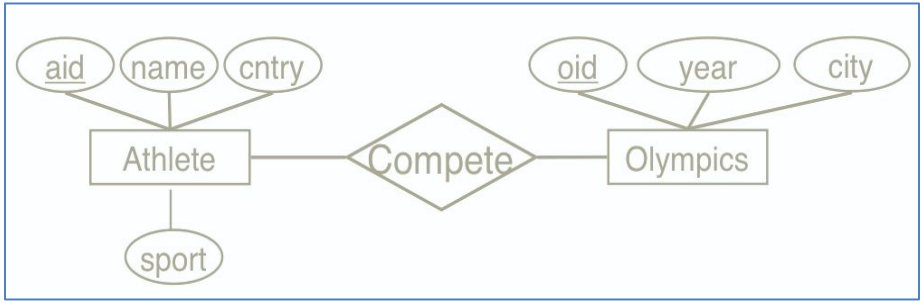
- A **key** for a relation R is a minimal set of attributes $A_1, \ldots, A_n$ that must be unique, i.e.,
  - no two tuples in (any instance of) R have the same key attributes $A_1, \ldots, A_n$.
- One key designated as primary key. Others are candidate keys
- Super key: Only uniqueness requirement, but no minimal requirement. Note: Every key is a superkey.

Examples:

- {aid} is the primary key in the Athlete relation
- {StudentID} is the primary key for Students relation
- {uniqname} is a candidate key for Students relation

Primary Key Constraint

Two ways to specify a primary key constraint:



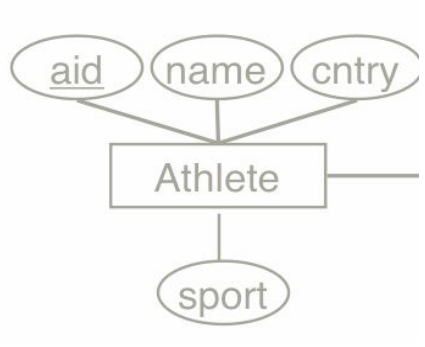

```
CREATE TABLE Athlete
(aid INTEGER PRIMARY KEY,
 name CHAR(30),
 country CHAR(20),
 sport CHAR(20));

CREATE TABLE Athlete
(aid INTEGER,
 name CHAR(30),
 country CHAR(20),
 sport CHAR(20),
 PRIMARY KEY(aid));
```

Not Null Constraint

- Example: Disallow null values for name attribute:

```
CREATE TABLE Athlete
(aid INTEGER PRIMARY KEY,
 name CHAR(30) NOT NULL,
 country CHAR(20),
 sport CHAR(20));
```



- **NULL** value implies the value is **unknown** or **inapplicable.** Semantics:
- Country and sport can be NULL (unspecified)
- But, name must be specified.

Primary Keys Properties

- SQL Standard:
  - A primary key attribute cannot be null
- Primary key often an integer ID for efficiency

Design consideration: Primary keys should be long-term stable, unique, and, ideally, not sensitive. Avoid addresses, SSN, etc.

Candidate Keys

- Candidate keys specified using UNIQUE.
- One of the candidate keys is chosen as the *primary key*.

```
CREATE TABLE Athlete
    (aid INTEGER,
     name CHAR(30) NOT NULL,
     country CHAR(20),
     sport CHAR(20),
    UNIQUE (name, country),
    PRIMARY KEY (aid));
```

- UNIQUE attributes can be NULL, unless NOT NULL is specified
  - Only the NON-NULL attribute values need to be UNIQUE.

Foreign Keys in SQL

- Compete table only has relationships between athletes in the Athlete table and the olympics games in the Olympics Table.

```
CREATE TABLE Compete
    (aid INTEGER,  oid INTEGER,
      PRIMARY KEY  (aid, oid)


    );
```

Foreign Keys in SQL

- Compete table has relationships only between athletes in the Athlete table and the olympics games in the Olympics Table.

```
CREATE TABLE Compete
    (aid INTEGER,  oid INTEGER,
      PRIMARY KEY  (aid, oid),
      FOREIGN KEY (aid) REFERENCES Athlete,
      FOREIGN KEY (oid) REFERENCES Olympics
    );
```

Foreign Keys in SQL

- Compete table has relationships only between athletes in the Athlete table and the olympics games in the Olympics Table.

```
CREATE TABLE Athlete
(aid INTEGER PRIMARY KEY,
 name CHAR(30),
 country CHAR(20),
 sport CHAR(20));
```

```
CREATE TABLE Compete
    (aid INTEGER,  oid INTEGER,
      PRIMARY KEY  (aid, oid),
      FOREIGN KEY (aid) REFERENCES Athlete,
      FOREIGN KEY (oid) REFERENCES Olympics
    );
```

Enforcing foreign key constraints implies referential integrity (no dangling references) is achieved.

Sample Database

- Download from course schedule: athlete_create.sql, athlete_drop.sql

- You can load it into Sqlite3 as follows:

```
% sqlite3 athlete.db
.read athlete_create.sql
```

Try a SQL query to print a table, e.g.,

```
SELECT * FROM Athlete;
```

Some useful Sqlite3 directives

```
.headers on
.mode column
.schema
.help
.read filename
.quit
```

- DuckDB uses same commands as Sqlite3. Download and try it out
- Oracle sqlplus, uses the following to read a file:

```
START athlete_create.sql
```

Enforcing Integrity Constraints (ICs)

- Whenever we modify the database
  - the DBMS must check for violations of ICs

- Enforcing Domain, Primary Key, Unique ICs is straightforward
  - Reject offending UPDATE / INSERT command

Enforcing Referential Integrity

- If a Compete tuple is inserted with no corresponding Athlete aid:
  - Insert operation is REJECTED!

```
CREATE TABLE Compete
    (aid INTEGER,  oid INTEGER,
      PRIMARY KEY  (aid, oid),
      FOREIGN KEY (aid) REFERENCES Athlete,
      FOREIGN KEY (oid) REFERENCES Olympics);
```

Enforcing Referential Integrity

- If a Compete tuple is inserted with no corresponding Athlete aid:
  - Insert operation is REJECTED!

- What if an Athlete tuple is deleted?

Referential Integrity enforced in SQL standard on DELETE/UPDATE

- NO ACTION (also called RESTRICT): disallow action (default)
- CASCADE: also delete all referencing tuples
- SET NULL / SET DEFAULT: sets foreign key value of referencing tuple to NULL or a default value

```
CREATE TABLE Compete
    (aid INTEGER,  oid INTEGER,
      PRIMARY KEY  (aid, oid),
      FOREIGN KEY (aid)
    REFERENCES Athlete
    ON DELETE CASCADE
    ON UPDATE SET NULL);
```

What happens to the Compete relation if we add a new athlete (with a new ID) to the Athlete table?

Where do ICs Come From?

- Based on real-world enterprise being modeled
- An IC is a statement about all possible instances!
- 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.
- Key and foreign key ICs are the most common

Destroying & Altering Relations

- To destroy the relation Olympics.
  - Schema information and tuples are deleted

```
DROP TABLE Olympics
```

- To alter the Athlete schema by adding a new column

```
ALTER TABLE Athlete
    ADD COLUMN age: INTEGER
```

- What do we put in the new field?
  - A null value: ‘unknown’ or ‘inapplicable’

Relational Model: Summary

Students

sid	name	login	age
13	Lisa	lsimp	40
41	Bart	bart	20

Courses

cid	cname	Cr.
E-484	EECS484	4
E-584	EECS584	3

Enrolled

sid	cid	Grade
41	E-484	A-
13	E-584	A+

- A tabular representation of data
- Simple and intuitive
- The most widely used database model by far
- Integrity constraints can be specified by the DBA, based on application semantics. DBMS checks for violations.
  - Two important ICs: primary and foreign keys
  - We always have domain constraints

    e.g. `INTEGER` fields must always contain integer values
- Views can be used for external schemas, and provide logical data independence