Relational models

Lecture 5

Wednesday - Sep 13, 2023

Housekeeping

- Project deliverable 3 (video) due Friday.
- Check your grades to verify that we've got it all correct!

Module	Week	Date	Day	Lectures/Quizzes	Deliverables/Notes
Relational Alg.	4	9/13	Wed	MTG6: L5 (Relational models)	
Relational Alg.	4	9/15	Fri		PrjDel 3 due (Topic proposal video)
Relational Alg.	5	9/18	Mon	MTG7: L6 (Relational Algebra 1)	
Relational Alg.	5	9/20	Wed	MTG8: L7 (Relational Algebra 2)	
Relational Alg.	5	9/24	Sun		HW4 due (Relational Algebra Study Guide)
Normal forms	6	9/25	Mon	MTG9: Quiz 2 today (Relational Algebra)	
Normal forms	6	9/27	Wed	MTG10: L8 (Analysis and Normal Forms 1)	
Normal forms	7	10/2	Mon	MTG11: L9 (Analysis and Normal Forms 2)	

Relational Models

Relational Model - Entities

- **Tables** correspond with entities and represent the fundamental object of manipulation.
- Tables have multiple columns called **fields**, each with unique names and data types. **Fields** correspond to *attributes* in our ER model.
- Tables have multiple rows called **records** that represent individual instances of the data stored in the table. These are sometimes called *tuples*.
- Relationships between tables are documented as **keys**. Keys are simply fields in a table, designated as keys. Keys have special properties and the database keeps track of them differently.
- A complete definition entities, attributes and relations, along with any data types and constraints is called a **schema**.

```sql
CREATE TABLE students (
    student_id INT PRIMARY KEY,
    name VARCHAR(100),
    year INT
);

CREATE TABLE courses (
    course_id INT PRIMARY KEY,
    title VARCHAR(100),
    credit_hours INT
);

CREATE TABLE student_courses (
    student_id INT,
    course_id INT,
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (course_id) REFERENCES courses(course_id),
    PRIMARY KEY (student_id, course_id)
);
```

In the example above, we define two entities: students and courses, and establish a many-to-many relationship between them. The relationships are documented in a separate table with foreign keys. The individual rows in the student-courses table represent pairs of students and courses or “student-course tuples”.

Relational Model - Attributes

Attributes

- Attributes are represented as fields or columns.
- A **domain** is the set of allowed values for each field.
- Field values are (normally) required to be atomic (indivisible).
- A special value **NULL** is a member of every domain.
- The DB designer must specify whether NULL is allowed or not (NOT NULL)
- Attributes take on specific data types. A database DDL specifies the data types:
    - Strings (VARCHAR, CHAR)
    - Numeric (NUMBER, INTEGER, FLOAT)
    - Date (DATE, DATETIME)
    - Objects (BFILE)

Relation Notation

Tables or entities can be represents in **relation notation**,

- *relationName (ID, Att1, �, AttN)*

For example:

- student(id, name, year)
- course(id, title, credit_hours)
- student_course(id, student_id, course_id)

We can also add modifiers to individual attributes:

- student(id [PK], name, year)
- course(id [PK], title, credit_hours)
- student_course(id [PK], student_id[FK], course_id[FK])

Relational Model - Attribute example with data

Questions

- What is the domain for each field?
- Can you write the relation set for each table?
- Can you draw a Chen diagram for these tables?
- Can you draw a Crow’s Foot diagram for these tables?
- Does the column order matter?

Students		
name	**year**	**id**
Amara Singh	1991	1
Youssef El-Mohamed	1987	2
Chen Wei	1978	3
Olga Petrova	NULL	4
Kwame Nkrumah	1965	5
Luc�a Gonz�lez	1980	6

Courses		
id	**name**	**hours**
1	CMSC391	3
2	CMSC475	3
3	CMSC508	3

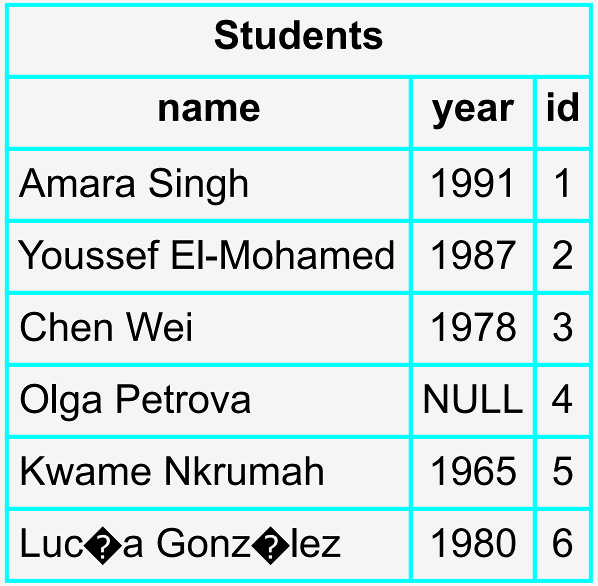
Relational Model - Keys

- A **key** can be any field (or fields) in a table.
- A key is used to **uniquely** identify each tuple (or row or record).
- Keys can take on any data type.
- A record can have multiple keys, as long as they uniquely identify the record.
  - A *super key* is a key of any size (one or more fields)
  - A *candidate key* is a key of minimal size (minimal number of fields)
  - A *primary key* is selected by the designer from one of the *candidate key* to become the primary identifier of the tuples in the relation
- Any two tuples in the relation are prohibited from having the same value on the key attributes at the same time
- *Primary keys* are typically size 1 but not necessarily.
- Every relation must have a primary key
- Keys are selected based on the intrinsic definition of the attributes, NOT on the existing values at any point in time

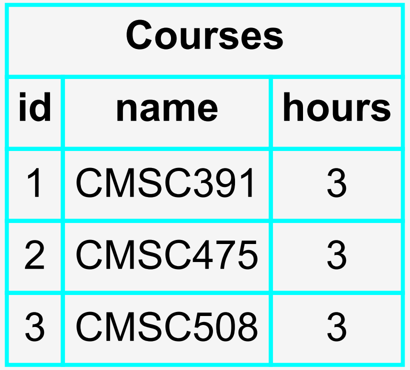
Relational Model - Key example with data

Questions

- What columns can uniquely identify each table?
- Can you identify *super keys* in each table?
- Can you identify *candidate keys* in each table?
- What is an appropriate *primary key* for each table?
- Would student.name be an appropriate key?



Students		
name	**year**	**id**
Amara Singh	1991	1
Youssef El-Mohamed	1987	2
Chen Wei	1978	3
Olga Petrova	NULL	4
Kwame Nkrumah	1965	5
Luc�a Gonz�lez	1980	6



Courses		
id	**name**	**hours**
1	CMSC391	3
2	CMSC475	3
3	CMSC508	3

Relational Model - Relationships

Relationships between tables and rows are modeled by connecting individual records using their *keys*.

A *foreign key* is a column(s) in a table that contains copies of the *keys* from another table referencing the related rows.

A *foreign key* must contain valid references to existing records in an existing table or must be null.

An attribute in a table can be simultaneously the primary key for that table and a foreign key referencing a column of another table

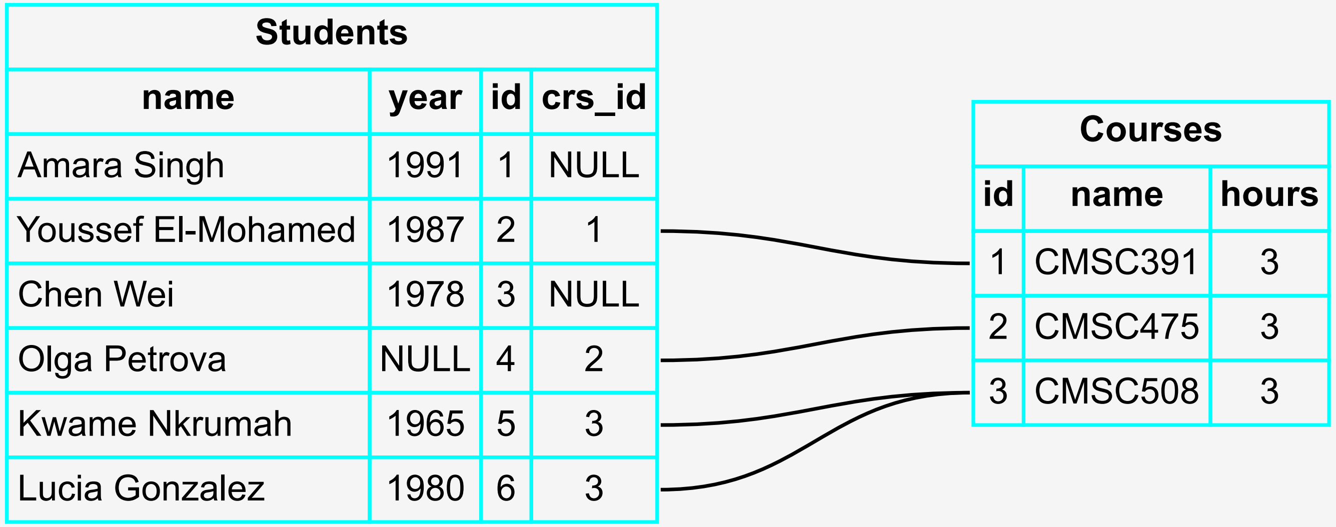
Attributes which are FKs do not need to have the same name as the referenced key.

Cardinalty and participation are represented through *foreign keys* and *integrity constraints*.

Relational Model - Relationship example with data

Questions

- What changed in the students table?
- What are the primary keys in each table?
- What is the column labled “crs_id”?
- What is the cardinality and participation of these tables?



Students

name	year	id	crs_id
Amara Singh	1991	1	NULL
Youssef El-Mohamed	1987	2	1
Chen Wei	1978	3	NULL
Olga Petrova	NULL	4	2
Kwame Nkrumah	1965	5	3
Lucia Gonzalez	1980	6	3

Courses

id	name	hours
1	CMSC391	3
2	CMSC475	3
3	CMSC508	3

Relational Model - Integrity Constraints

An *integrity constraint* is:

- a condition that must be true for every instance (row, record, tuple) in the relation (table).
- specified in the definition of the database schema
- verified by the DBMS at every CRUD operation (create, read, update, delete)

Domain integrity

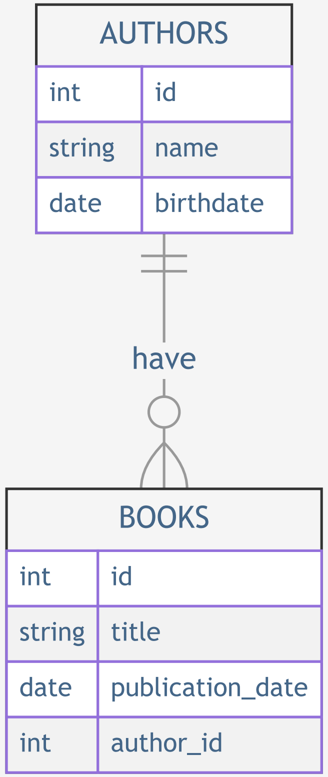
- ensure fields always contain the correct data
- by data type
- by length or size
- by negative values
- by closed set of allowed values
- by NULL or NOT NULL values
- by UNIQUE or not

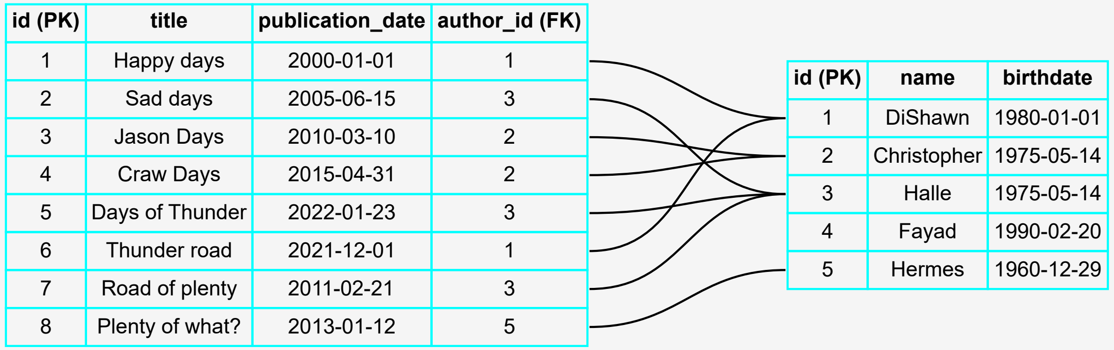
Referential integrity

- maintains cardinality and participation
- primary keys must be unique
- foreign keys must exist

Using *integrity constraints* the DBMS will maintain a perfect state for your database. By specifying integrity constraints, a designer lets the DBMS do the work for them.

Relational Model - Relationship example with data





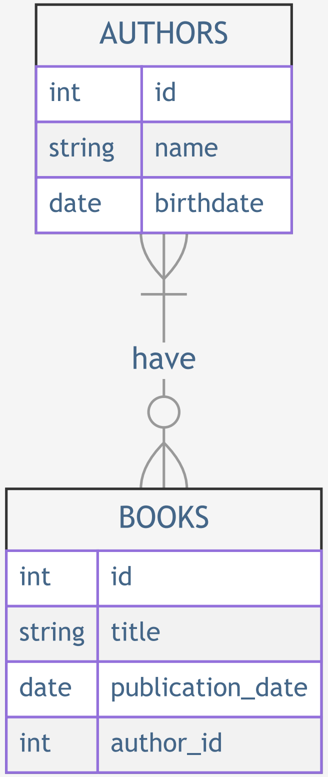
id (PK)	title	publication_date	author_id (FK)
1	Happy days	2000-01-01	1
2	Sad days	2005-06-15	3
3	Jason Days	2010-03-10	2
4	Craw Days	2015-04-31	2
5	Days of Thunder	2022-01-23	3
6	Thunder road	2021-12-01	1
7	Road of plenty	2011-02-21	3
8	Plenty of what?	2013-01-12	5

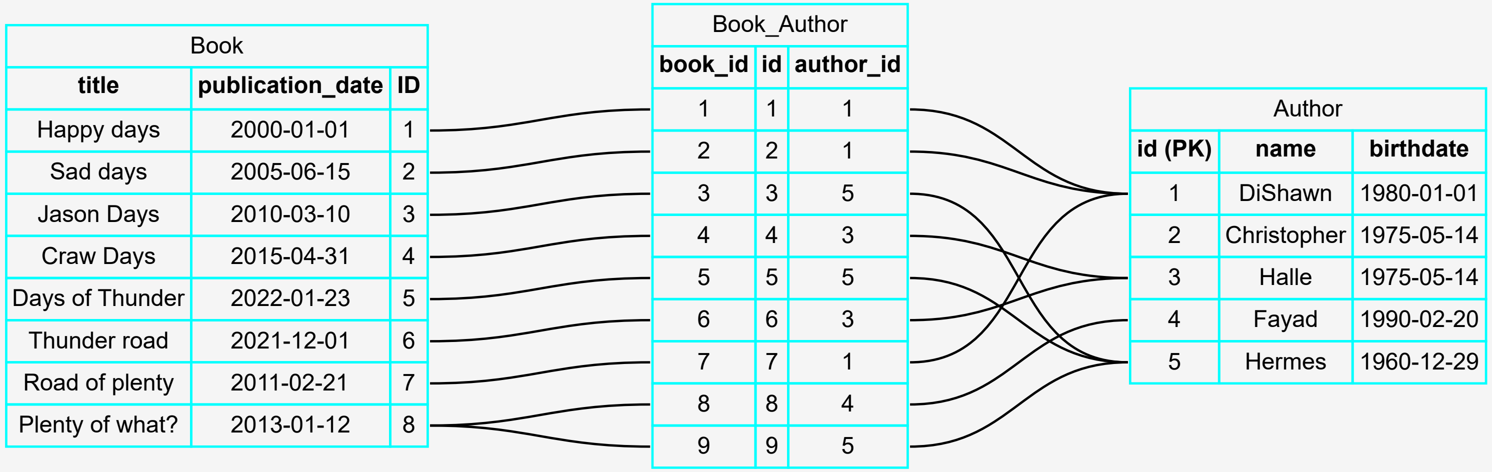
id (PK)	name	birthdate
1	DiShawn	1980-01-01
2	Christopher	1975-05-14
3	Halle	1975-05-14
4	Fayad	1990-02-20
5	Hermes	1960-12-29

Questions

- Does the ER diagram on the left match the tables on the right?
- How do we ensure partial or total participation?
- Write the detailed relation sets for this model.
- How would we modify the tables to create a many-to-many relationship?

Relational Model - Relationship example with data





Book

title	publication_date	ID
Happy days	2000-01-01	1
Sad days	2005-06-15	2
Jason Days	2010-03-10	3
Craw Days	2015-04-31	4
Days of Thunder	2022-01-23	5
Thunder road	2021-12-01	6
Road of plenty	2011-02-21	7
Plenty of what?	2013-01-12	8

Book_Author

book_id	id	author_id
1	1	1
2	2	1
3	3	5
4	4	3
5	5	5
6	6	3
7	7	1
8	8	4
9	9	5

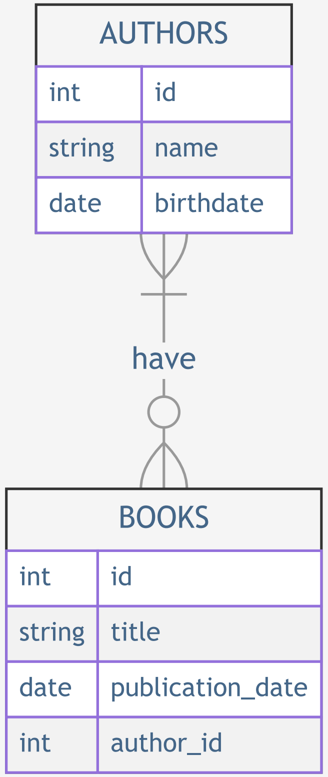
Author

id (PK)	name	birthdate
1	DiShawn	1980-01-01
2	Christopher	1975-05-14
3	Halle	1975-05-14
4	Fayad	1990-02-20
5	Hermes	1960-12-29

Detailed relation sets

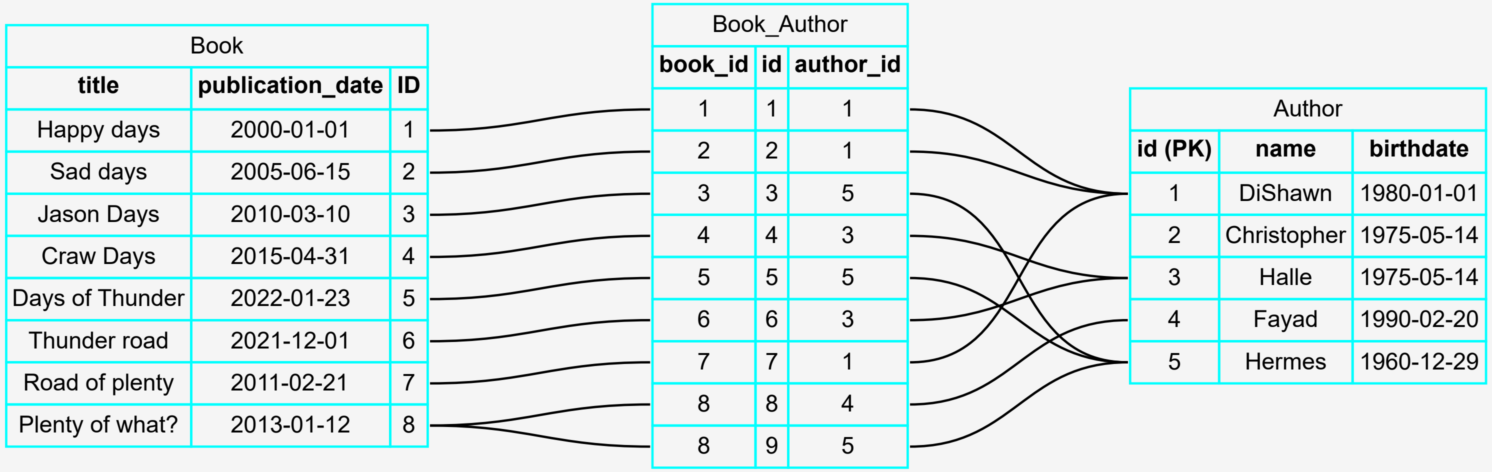
- book(id [PK], title [NOT NULL], publication_date [NOT NULL])
- author(id[PK], name [NOT NULL], birthdate [NOT NULL])
- book_author(id [PK], book_id [FK,NOT NULL], author_id [FK,NOT NULL])
  FK(book_id) references (book.id)
  FK(author_id) references (author.id)

Relational Model - Relationship example with data



AUTHORS	
int	id
string	name
date	birthdate

BOOKS	
int	id
string	title
date	publication_date
int	author_id



Book

title	publication_date	ID
Happy days	2000-01-01	1
Sad days	2005-06-15	2
Jason Days	2010-03-10	3
Craw Days	2015-04-31	4
Days of Thunder	2022-01-23	5
Thunder road	2021-12-01	6
Road of plenty	2011-02-21	7
Plenty of what?	2013-01-12	8

Book_Author

book_id	id	author_id
1	1	1
2	2	1
3	3	5
4	4	3
5	5	5
6	6	3
7	7	1
8	8	4
8	9	5

Author

id (PK)	name	birthdate
1	DiShawn	1980-01-01
2	Christopher	1975-05-14
3	Halle	1975-05-14
4	Fayad	1990-02-20
5	Hermes	1960-12-29

Questions

- How would you modify this to ensure that books exactly one author?
- How would you modify this to ensure that authors at least one book?
- How would you modify this to ensure that authors has exactly one book?
- Could you change these features in real-time?

Relational Model - Unanswered questions

How would I model a composite attribute?

A composite attribute is something like *full name*, for example *John Denes Leonard II*. You'll probably want to give each atomic piece of the *full name* it's own field/attribute and allow individual pieces to be NULL if they aren't used. THEN, you might consider setting *full name* to be a calculated field, so that it is computed whenever needed by combining the individual first, middle, last, and initial.

How would I model a multivalued attribute?

A multivalued attribute is something like *phone number*, that is, a *person* can have zero or more *phone numbers*. There are several solutions, either by adding a *person_id* foreign key to the *phone number* relation, OR by creating a separate link/join table (see the previous slide) and adjust the attributes (e.g., NOT NULL, UNIQUE, etc.) to get the cardinality and participation you're after.

How would I model a weak entity?

A weak entity is one that depends on a strong entity to survive. That is, if the *parent strong entity* is deleted, the weak entity is no longer needed. This is modeled using a *TRIGGER* in the *schema*. When the string entity is deleted, all corresponding weak entities are deleted, too.

Housekeeping

- Project deliverable 3 (video) due Friday.
- Check your grades to verify that we’ve got it all correct!

Module	Week	Date	Day	Lectures/Quizzes	Deliverables/Notes
Relational Alg.	4	9/13	Wed	MTG6: L5 (Relational models)	
Relational Alg.	4	9/15	Fri		PrjDel 3 due (Topic proposal video)
Relational Alg.	5	9/18	Mon	MTG7: L6 (Relational Algebra 1)	
Relational Alg.	5	9/20	Wed	MTG8: L7 (Relational Algebra 2)	
Relational Alg.	5	9/24	Sun		HW4 due (Relational Algebra Study Guide)
Normal forms	6	9/25	Mon	MTG9: Quiz 2 today (Relational Algebra)	
Normal forms	6	9/27	Wed	MTG10: L8 (Analysis and Normal Forms 1)	
Normal forms	7	10/2	Mon	MTG11: L9 (Analysis and Normal Forms 2)	