The Relational Model

Dirk Van Gucht

The relational model

- The relational model (RM) is an approach for handling data through a structure and language that is consistent with first-order predicate logic,
- The purpose of the relational model is to provide a declarative method for specifying data and queries:
  - users state what information the database contains and what information they want from it,
  - the database management system software take care of describing data structures for storing the data and retrieval procedures for answering queries.

The relational model

- Most relational databases use the SQL data definition and query language;
- These DBMS implement what can be regarded as an engineering approximation to the relational model.
  - A table in a SQL database schema corresponds to a predicate variable;
  - key constraints, other constraints, and SQL queries correspond to predicates.
- The central idea of a relational model is to describe a database as a collection of predicates over a finite set of predicate variables, describing constraints on the possible values and combinations of values.

The relational model

- The fundamental assumption behind a relational model is that all data is represented as mathematical *n*-ary relations,
  - n is the number of domains, where each domain is associated with an attribute
- An *n*-ary relation being a subset of the Cartesian product of n domains.
- In the mathematical model, reasoning about such data is done in two-valued predicate logic (true/false)
- Only those n-tuples appear in a relation whose values satisfy some predicate

The relational database

- A relational database is a collection of relations (tables).

Student

Sid	Sname	Major	Byear
s1	John	CS	1990
s2	Ellen	Math	1995
s3	Eric	CS	1990
s4	Ann	Biology	2001

Course

Cno	Cname	Dept
c1	Dbs	CS
c2	Calc1	Math
c3	Calc2	Math
c4	AI	Info

Enroll

Sid	Cno	Grade
s1	c1	B
s1	c2	A
s2	c3	B
s3	c1	A
s3	c2	C

Terminology: definitions

Attributes	relation column names
Relation schema	consists of a relation name and a set of attributes
Tuples	rows of a relation
Tuple components	a tuple has one component for each attribute of the relation; a tuple component value must be from the domain of the attribute
Relation instance	a finite set of tuples
Relation	a pair of a relation schema and a relation instance

- The schema of a relation is called its meta-data.
- The instance of a relation is called its data.

Terminology: example

Student ← relation name

Sid	Sname	Major	Byear	
				← attributes
s1	John	CS	1990	← tuple
s2	Ellen	Math	1995	← tuple
s3	Eric	CS	1990	← tuple
s4	Ann	Biology	2001	← tuple

- The relation name and header of the table is the relation schema.
- The body of the table is the relation instance.

Attribute domains

- Different types are possible:
  - Basic types: boolean, integer, real, character, text;
  - Composite types: monetary, date;
  - Enumeration types: list of values, sets, arrays;
  - Semi-structured types: XML, JSON;
  - Arrays,

and many more!

Tuples and tuple components

- Consider the Student(Sid, Sname, Major, Byear) schema and the tuple t = (s1, John, CS, 1990).
- Then t.Sid, t.Sname, t.Major, and t.Byear are the tuple components of t and their domain values are s1, John, CS, and 1990, respectively.

Null values

- NULL represents a missing or an unknown value of a tuple (incomplete information).
- E.g., (s5, Marc, NULL, 1995) indicates that the Major component of this tuple is missing or unknown.
- Caveat: multiple occurrences of NULL values do not imply that they are the same: they may or may not be equal.
- E.g., (s6, NULL, NULL, 2000) indicates that the Name and Major components of this tuple are missing or unknown. If both are missing, they obviously need not represent the same value!

Keys: definition

- A subset of attributes of the schema of a relation is a key if we do not allow two different tuples in any of its possible relation instances to have the same values across the attributes of the key.
- A key of a relation is a constraint of that relation.

Keys: example

- The key attributes of each relation are underlined.

Student

Sid	Sname	Major	Byear
s1	John	CS	1990
s2	Ellen	Math	1995
s3	Eric	CS	1990
s4	Ann	Biology	2001

Course

Cno	Cname	Dept
c1	Dbs	CS
c2	Calc1	Math
c3	Calc2	Math
c4	AI	Info

Enroll

Sid	Cno	Grade
s1	c1	B
s1	c2	A
s2	c3	B
s3	c1	A
s3	c2	C

Keys: example

- The key attributes of each relation are underlined.

Student

<u>Sid</u>	Sname	Major	Byear
s1	John	CS	1990
s2	Ellen	Math	1995
s3	Eric	CS	1990
s4	Ann	Biology	2001

Course

<u>Cno</u>	Cname	Dept
c1	Dbs	CS
c2	Calc1	Math
c3	Calc2	Math
c4	AI	Info

Enroll

<u>Sid</u>	<u>Cno</u>	Grade
s1	c1	B
s1	c2	A
s2	c3	B
s3	c1	A
s3	c2	C

Keys: example (continued)

- Major is not a key for Student, since two different tuples share the value “CS” for this attribute.

Student

Sid	Sname	Major	Byear
s1	John	CS	1990
s2	Ellen	Math	1995
s3	Eric	CS	1990
s4	Ann	Biology	2001

Keys: example (continued)

- Sid alone is not a key for Enroll. Also, Cno alone is not a key for Enroll.

Enroll

Sid	Cno	Grade
s1	c1	B
s1	c2	A
s2	c3	B
s3	c1	A
s3	c2	C

Keys: significance

- Each key value of a tuple acts as a unique name (reference, identifier) for that tuple.
- Consequently, when a key value appears somewhere in the database, we can equivalently think of the appearance of the referenced tuple at that place in the database.

Enroll

	Sid	Cno	Grade
s1,c1 →	s1	c1	B
s1,c2 →	s1	c2	A
s2,c3 →	s2	c3	B
s3,c1 →	s3	c1	A
s3,c2 →	s3	c2	C

Keys: significance (continued)

Consequently, when a key value appears somewhere in the database, we can equivalently think of the appearance of the referenced tuple at that place in the database.

Enroll

<u>Sid</u>	<u>Cno</u>	Grade
s1 (John, CS, 1990)	c1(Dbs, CS)	B
s1 (John, CS, 1990)	c2(Calc1, Math)	A
…		

Keys: significance (continued)

- In contrast, a non-key value references a set of tuples; each tuple in this set can of course be identified by its key value.
- Example: the Major value "CS" references the set of tuples identified by the key values "s1" and "s3".

Student

<u>Sid</u>	Sname	Major	Byear
s1	John	CS	1990
s2	Ellen	Math	1995
s3	Eric	CS	1990
s4	Ann	Biology	2001

The language SQL

- SQL is a language to handle relations in the relational model. We distinguish:
  - The SQL Data Definiton Language (DDL), which is used to define define database and relation schemas;
  - The SQL Data Manipulation Language (DML), which is used to define, modify, and query relation instances.

SQL DDL: defining database schemas

- We define the database schema simply by specifying the statement:

```
CREATE DATABASE  University;

\connect University
Other SQL statements
\q
```

SQL DDL: defining relation schemas

- We define the relation schemas of the three relations in our running example:

Student(<u>Sid</u>, Sname, Major, Byear)

```
CREATE TABLE Student
    (Sid       TEXT,
     Sname VARCHAR(30),
     Major   VARCHAR(15),
     Byear    INTEGER,
     PRIMARY KEY (Sid)
    );
```

Course(<u>Cno</u>, Cname, Dept)

```
CREATE TABLE Course
    (Cno      TEXT,
     Cname VARCHAR(20),
     Dept     VARCHAR(15),
     PRIMARY KEY (Cno)
    );
```

Enroll(<u>Sid</u>, <u>Cno</u>, Grade)

```
CREATE TABLE Enroll
    (Sid      TEXT,
     Cno      TEXT,
     Grade VARCHAR(2),
     PRIMARY KEY (Sid, Cno)
    );
```

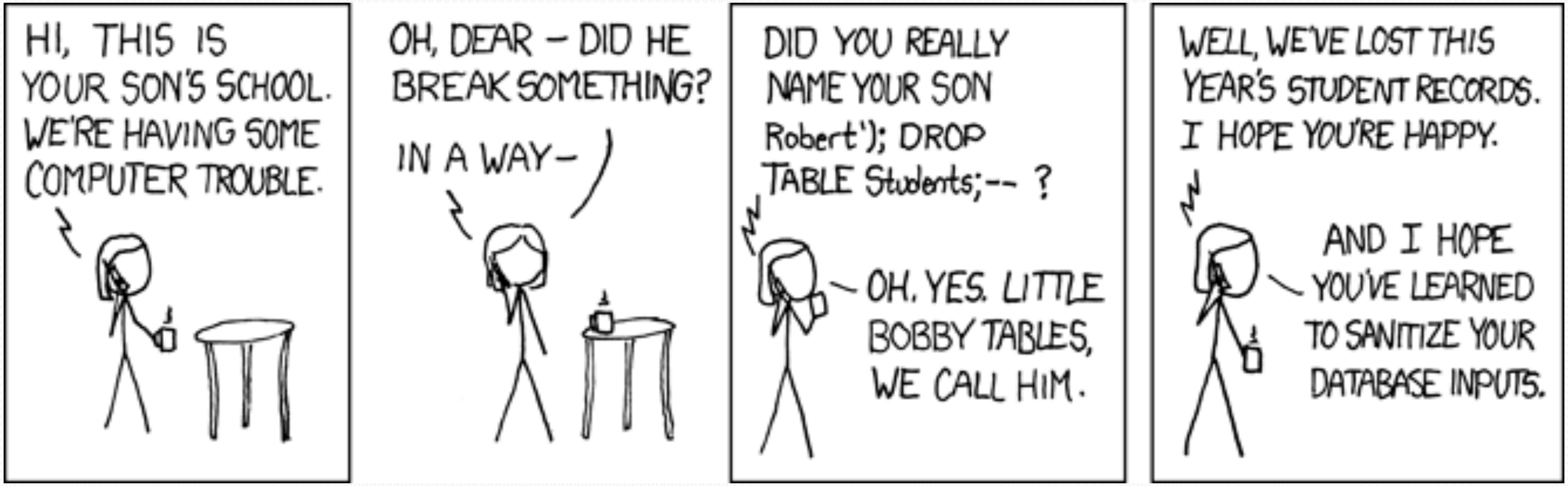
SQL DDL: NOT NULL constraints

- In SQL, it is possible to disallow null values for certain attributes.
- For obvious reasons, it is advisable to disallow null values for key attributes.
- Example:

```
CREATE TABLE Student
    (Sid        TEXT NOT NULL,
     Sname VARCHAR(30),
     Major   VARCHAR(15),
     Byear    INTEGER,
     PRIMARY KEY (Sid)
    );
```

SQL DDL: other operations

- The SQL statement DROP DATABASE is used to remove a database.
- E.g., DROP DATABASE University removes the database University
- The SQL statement DROP TABLE is used to remove a relation.
- E.g., DROP TABLE Student removes the relation Student.


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

SQL DML: defining relation instances

- The SQL statement INSERT INTO is used to populate relations.
- Example:

```
INSERT INTO Student VALUES('s1', 'John', 'CS', 1990);
INSERT INTO Student VALUES('s2', 'Ellen', 'Math', 1995);
INSERT INTO Student VALUES('s3', 'Eric', 'CS', 1990);
INSERT INTO Student VALUES('s4', 'Ann', 'Biology', 2001);
```

creates

Student

Sid	Sname	Major	Byear
s1	John	CS	1990
s2	Ellen	Math	1995
s3	Eric	CS	1990
s4	Ann	Biology	2001

SQL DML: defining relation instances (alternative)

- The SQL statement INSERT INTO is used to populate relations.
- Alternative:: INSERT INTO Student VALUES(‘s1’, ‘John’, ‘CS’, 1990),
(‘s2’, ‘Ellen’, ‘Math’, 1995),
(‘s3’, ‘Eric’, ‘CS’, 1990),
(‘s4’, ‘Ann’, ‘Biology’, 2001);

also creates

Student

Sid	Sname	Major	Byear
s1	John	CS	1990
s2	Ellen	Math	1995
s3	Eric	CS	1990
s4	Ann	Biology	2001

- However, a fifth tuple insertion

INSERT INTO Student VALUES (‘s1’, ‘Linda’, ‘Math’, 1993);

will be rejected, because (‘s1’, ‘John’, ‘CS’, 1990) is already in the Student relation, and therefore the new insertion would violate the primary key constraint on Sid.

SQL DML: retrieving relation instances

- To retrieve a relation instance, use the SQL statement SELECT FROM.
- Example:

```
SELECT * FROM Student;
SELECT * FROM Course;
SELECT * FROM Enroll;
```

retrieves the relation instances of our running example.

SQL DML: querying

- A single relation can be queried by selectively retrieving parts of that relation based on a condition. (SELECT FROM WHERE)
- Example:

```
SELECT S.sid, S.sname
FROM   Student S
WHERE S.major = 'CS';
```

retrieves sid and name of each student majoring in CS:

Sid	Sname
s1	John
s3	Eric

SQL DML: querying (continued)

- Multiple relations can be queried by linking (joining) them.
- Example:

```
SELECT S.sid, S.sname, E.cno
FROM   Student S, Enroll E
WHERE S.sid = E.sid AND E.grade = 'B';
```

returns each (sid, sname, cno) tuple where student sid received a B in course cno.

Sid	Cno
s1	c1
s2	c3

SQL DML: querying (continued)

- Set operations (UNION, INTERSECT, EXCEPT) can be applied to multiple SQL queries
- Example:

```
(SELECT S.sid FROM Student S)
EXCEPT
(SELECT E.sid FROM Enroll E WHERE E.Grade = `A’);
```

selects the sid values of students who did not receive a `A’ grade in any of their courses

Sid
s2
s4

Referential integrity

- We already encountered primary keys, which impose constraints on single relations.
- Other constraints involve multiple relations: in the relation Enroll, the tuple (s1, c1, 'B') only makes sense because s1 references the student identified by this sid in the relation Student and c1 references the course identified by this cno.
- However, inserting the tuple (s5, c1, 'A') into Enroll makes no sense, since s5 does not reference any student in Student.

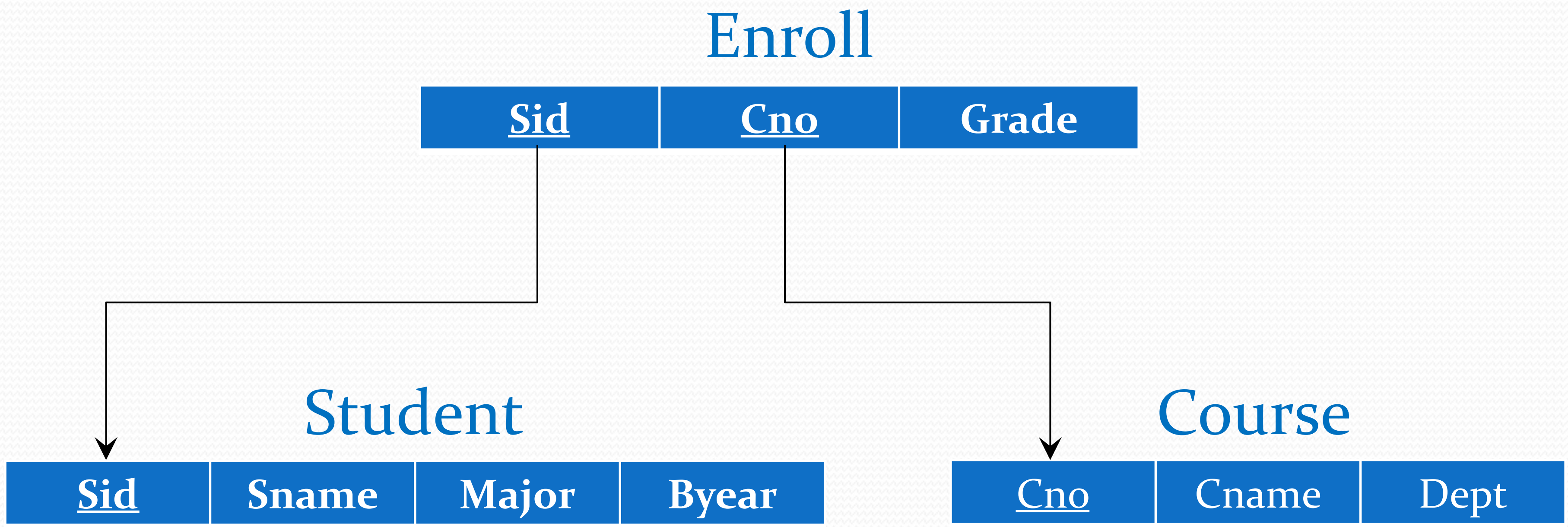
Foreign keys

- In the relation schemas

Student(<u>Sid</u>, Sname, Major, Byear)
Course(<u>Cno</u>, Cname, Dept)
Enroll(<u>Sid</u>, <u>Cno</u>, Grade)

the attribute Sid of Enroll is a foreign key referencing the primary key Sid of Student. Likewise, the attribute Cno of Enroll is a foreign key referencing the primary key Cno of Course.

- Graphically:



Foreign keys in SQL

- The foreign key constraints to Enroll can be added as follows:

```
CREATE TABLE Enroll
    (Sid     INTEGER,
     Cno     INTEGER
     Grade VARCHAR(2),
     PRIMARY KEY (Sid, Cno),
     FOREIGN KEY (Sid)   REFERENCES Student(SID),
     FOREIGN KEY (Cno) REFERENCES Course(Cno)
    );
```

Foreign keys and insertion

- Foreign keys impose an order on insertions in relations: the tuple (s1, c1, 'B') can only be inserted in Enroll after a tuple identified by the Sid value s1 exists in Student and after a tuple identified by the Cno value c1 exists in Course.
- Hence, inserting the tuple (s5, c1, 'A') in the current state of Enroll will be rejected, since there is no tuple in Student with Sid value s5.

Foreign keys and deletion

- Foreign keys also necessitate cascading deletions: deleting the tuple (s1, 'John', 'CS', 1990) from Student requires deleting all tuples in Enroll referencing the Sid value s1. (A similar effect occurs upon deleting a tuple from Course.)

Foreign keys and deletion (c’ed)

- Two option in SQL:
  - Allow cascading deletions.
  - Disallow cascading deletions (and hence refuse deletion).
- Example:

```
CREATE TABLE Enroll
    (Sid      INTEGER,
     Cno      INTEGER
     Grade VARCHAR(2),
     PRIMARY KEY (Sid, Cno),
     FOREIGN KEY (Sid)   REFERENCES Student(SID) ON DELETE CASCADE,
     FOREIGN KEY (Cno) REFERENCES Courses(Cno) ON DELETE RESTRICT
    );
```