CS2Bh: Current Technologies

Introduction to XML and Relational Databases

Spring 2005

The Relational Model

The relational model

Proposed by Codd in 1970. It is the dominant data model today.

Well-known relational DBMS: DB2, Oracle, Sybase, Ingres, Access, SQLServer, FoxBase.

In the relational data model we organize the data into tables. We don't (initially) worry about how these tables are implemented.

In the rest of the lectures we are going to discuss

- ✓ Relational databases:
  - Basics: relations, schemas
  - Integrity constraints: keys, foreign keys, referential integrity
- ✓ Relational query languages
  - Relational algebra
  - SQL

What is a relational database?

Students:

sid	sname	gpa
001	joe	3.0
002	mary	2.8
003	grace	4.0

Enroll:

sid	cid	grade
001	166	B
003	166	A
003	CS2	A

Courses:

cid	cname	credits	instructor
166	math	3	poe
CS2	db	4	fan

Question: can a table have duplicate rows?

Some terminology

- ✓ Column names: **attributes**, **fields**.
- ✓ Rows: **tuple, records**.
- ✓ Each attribute has values taken from a **domain**, e.g. the domain of sid is string, and the domain of gpa is real.

  Domains: *string, integer, real, date*, etc -- atomic types
- ✓ A table is called a **relation**, which is a set of tuples (records).
- ✓ The **cardinality** of a relation is the number of tuples in it.
- ✓ The **degree** (**arity**) of a relation is the number of fields.

Questions:

- ✓ What are the cardinality and arity of relation Students?
- ✓ Can we put a Student tuple and a Courses tuple in the same relation? If not, how to prevent this? In other words, how to describe relations?

Describing relations

A relation is defined by a (**relation**) **schema**, which specifies domains of each field in the relation.

Students (sid: string, sname: string, gpa: real)

Courses (cid: string, cname: string, credits: integer, instructor: string)

Enroll (sid: string, cid: string, grade: string)

- A (relation) schema is like a type, expressed in DDL (vs. type system). Therefore, it has many (possible) instances (values).
- A **relation** (**instance**) is a table (a set of tuples). Example: Students table.
- A (**relational**) **database schema** is a collection of (relation) schemas.
- A **relational database** (instance) is a collection of tables, each has a distinct name.

questions

Are the following correct?

- A relational database is a set of tuples.
- A schema can be viewed as a type. Then, a relation of the student schema

  Students (sid: string, sname: string, gpa: real)

is a value of the type.

Collection types

- Sets: unordered, no duplicates.

  e.g., {1, 2, 3} = {3, 2, 1} = {1, 3, 2, 2}
- Bags: unordered, duplicates count.

  e.g., {| 1, 2, 2, 3|} = {| 2, 3, 1, 2|}

  {| 1, 2, 3|} <> {|1, 2, 3, 3|}
- Lists: ordered, allowing duplicates.

  e.g.: [1, 2, 3] <> [2, 3, 1]

  [1, 2, 3, 3] <> [1, 2, 3]

Integrity constraints (ICs)

- ✓ **IC**: condition that must be true for any instance of the database.

  Example: a schema specifies domains of the fields in the relation, and is called *domain constraints*.
  - ICs are specified when schema is defined.

  Note: this might not be true in Web databases.
  - ICs are checked when relations are modified.
- ✓ A **legal** instance of a relation is one that satisfies all specific ICs.

Integrity constraints (cont'd)

DBMS does not allow illegal instances.

- ✓ If the DBMS checks ICs, stored data is more faithful to real-world meaning, that is, for consistency and accuracy.
- ✓ When ICs are enforced, DBMS also avoids data entry errors.

Remarks:

- ✓ For relational (and object-oriented) databases, domain constraints are a must! Contrast this with XML DTD
- ✓ There are many other common integrity constraints, which are given below.

key constraints

A set of fields is a (**candidate**) **key** for a relation if:

- ✓ no two distinct tuples can have same values in all key fields,
- ✓ any proper subset of the key does not satisfy this condition.

Example: sid is a key for Students.

Question: What about sname? What is a key of Enroll?

Students:

sid	sname	gpa
001	joe	3.0
002	mary	2.8
003	grace	4.0

Enroll:

sid	cid	grade
001	166	B
003	166	A
003	CS2	A

Primary key constraints

- If a set of fields satisfies condition 1 but does not satisfy condition 2, then it is called a **superkey** for the relation.
- If there are more than one (candidate) keys for a relation, one is chosen to be the **primary** key.

Example: sid is a primary key for Students. The set {sid, gpa} is a superkey.

Students:

sid	sname	gpa
001	joe	3.0
002	mary	2.8
003	grace	4.0

Enroll:

sid	cid	grade
001	166	B
003	166	A
003	CS2	A

More on primary keys

- There may be many candidate keys, but only one primary key.
- Given a relation schema, the primary key of the schema is underlined.

Students (sid: string, sname: string, gpa: real)

Enroll (sid: string, cid: string, grade: string)

Questions:

- Are the following claims correct?
  - Every relation has at least one key.
  - Every relation has one and only one primary key.
  - Every relation has one and only one superkey.

questions

- Recall that relation Courses is specified by the schema:

Courses(cid: string, cname: string, credits: integer, instructor:string)

What is the (reasonable) primary key of this relation?

- Can we insert a tuple (001, john, 4.0) into Students?

sid	sname	gpa
001	joe	3.0
002	mary	2.8
003	grace	4.0

Foreign keys, referential integrity (1)

- **Foreign key**: A set of fields in one relation that is used to 'refer' to a tuple in another relation. This set must correspond to the primary key of the second relation.

Example: sid is a foreign key in Enroll referring to Students.

Students (sid: string, sname: string, gpa: real)

Enroll (sid: string, cid: string, grade: string)

Students:

sid	sname	gpa
001	joe	3.0
002	mary	2.8
003	grace	4.0

Enroll:

sid	cid	grade
001	166	B
003	166	A
003	CS2	A

Foreign keys, referential integrity (2)

- ✓ Understanding: foreign keys vs. logical pointers.
- ✓ Foreign key constraints: Let S1 be a foreign key in relation R1 referring to R2. Then the values of S1 in R1 must match (be a subset of) the values of S1 in R2.

Question: does the following satisfy the foreign key constraint?

Students:

sid	sname	gpa
001	joe	3.0
002	mary	2.8
003	grace	4.0

Enroll:

sid	cid	grade
002	166	B
003	166	A
005	CS2	A

Foreign keys, referential integrity (3)

- ✓ **Referential integrity**: the database does not include any invalid foreign key value. That is, all foreign key constraints are enforced.

Questions:

- ✓ Why foreign key constraints?

This is analogous to the dangling pointer problem in programming languages.

- ✓ Can you name a data model without referential integrity?

Links in HTML.

Enforcing referential integrity

Recall that sid in Enroll is a foreign key that references Students.

- What should be done if an Enroll tuple with a non-existent student sid is inserted?

  Reject it!

- What should we do if a Students tuple is deleted?
  - Also delete all Enroll tuples that refer to it (CASCADE)
  - Disallow deletion of a Students tuple that is referred to (NO ACTION)
  - Set sid in Enroll tuples that refer to it to a default sid (SET DEFAULT).
- What should we do if primary key of Students tuple is updated?

  same as the deletion case

Example

Recall that sid is the primary key of Students, and sid in Enroll is a foreign key that references Students.

Students:

sid	sname	gpa
001	joe	3.0
002	mary	2.8
003	grace	4.0

Enroll:

sid	cid	grade
001	166	B
003	166	A
003	CS2	A

Question: what should we do when we

- insert (001, joe, 3.8) into Students?
- insert (004, CS2, A) into Enroll?
- delete (001, joe, 3.0) from Students?
- modify 001 to 004 in Students?

Where do ICs come from?

- ICs are based upon the semantics of the real-world enterprises that is being described in the database relations.
- We can check a database instance to see if an IC is violated, but we can't infer if an IC is true by looking at an instance.

An IC is a statement about all possible instances!

- Key and foreign key constraints are the most common ones; more general ICs are supported too.
- Why ICs? Database design, query optimization, updates, and much more!

Summary – what you should remember!

- What is a relational database? Tuple? Attribute? Field? Domain? Schema? Database schema? Instance?
- What are integrity constraints? Why study them? How to decide whether an instance is legal? Where do ICs come from? What integrity constraints are a must for relational databases? Name two other common ICs.
  - Primary key constraints. What? Why?
  - Questions: Are the following claims correct?
    - Every relation has at least one key.
    - Every relation has one and only one primary key.
    - Every relation has one and only one superkey.
  - Foreign key constraints. What? Why?