Database Security and Reliability

Tom Kelliher, CS 325

Nov. 9, 2011

1 Administrivia

Announcements

Assignment

Read 6.4–5.

From Last Time

Assurance.

Outline

1. Introduction.
2. Reliability and integrity.

Coming Up

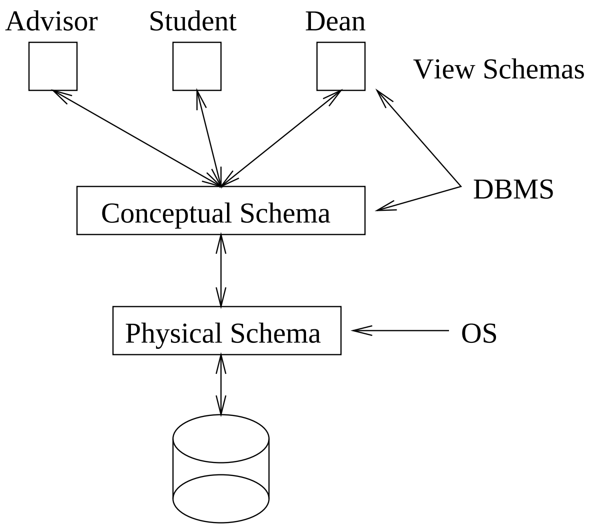
Sensitive data and inference in databases.

2 Introduction

What is:

1. a database?
2. a DBMS?

   Examples: Oracle, Sybase, MS SQL Server, MySQL, PostgreSQL.



3. SQL?
4. a Transaction? (Changes state of database.)
5. a Transaction Processing System? (Transactions, TP monitor, DBMS, database.)

   Potentially multiple distributed databases on heterogeneous platforms.

Database characteristics. Envision an airline reservations system.

1. Availability.
2. Reliability.
3. Throughput.

4. Response time.

5. Lifetime.

6. Integrity at the levels: physical database, logical database, element.

7. Audits. Sarbanes-Oxley.

8. Authentication and access control.

2.1 Relational Databases

1. Table, relation.

   View as a predicate — a statement of truth. A set.

2. Row, tuple.

   An ordered n-tuple.

3. Column, attribute.

   Properties of tuples.

4. Domain of an attribute.

Example relations:

```
CREATE TABLE "student" (
   "id" integer NOT NULL,
   "name" character(20) NOT NULL,
   "address" character(50),
   "status" character(10) DEFAULT 'Freshman',
   Constraint "stu_key" Primary Key ("id")
);

CREATE TABLE "transcript" (
   "stuid" integer,
   "crscode" character(6),
   "semester" character(6),
   "grade" character(1),
   CONSTRAINT "gradecheck" CHECK ( grade in ('A', 'B', 'C', 'D', 'F')),
   CONSTRAINT "stuidcheck" CHECK (stuid > 0 AND stuid < 1000000000)
);
```

psql demo and sample queries:

```
-- Get name of student with particular Id #.
select Name
from Student
where Id = '987654321';


-- Get Id and Name of all seniors.
select Id, Name
from Student
where Status = 'Senior';


-- Get Name, Course, and Grade for all seniors.
-- Must match tuples (join) between two relations.
select Name, CrsCode, Grade
from Student, Transcript
where StuId = Id and Status = 'Senior';
```

2.2 Properties of Transactions

Some integrity constraints for Student Registration System:

1. Student Ids are unique.
2. Students must satisfy course prerequisites before registering for a course.
3. The number of students registered for a course cannot exceed the course cap.
4. Suppose there are two ways to count the number of students registered for a course (aggregate on Transcript relation and attribute of Courses relation). These two ways of counting must yield the same result.

A transaction's ACID properties:

1. Atomicity: All or nothing.
2. Consistency: Integrity constraints are preserved.

   Transaction designer assumes database is initially consistent.
3. Isolation: Consider multiple simultaneous transactions. What bad things can happen?
   (a) Serial execution.

   (b) Transaction schedules: serial, concurrent.

   (c) Serializable concurrent schedules.

   (d) Isolation definition: *Even though transactions are executed concurrently, the overall effect of the schedule must be the same as if the transactions had executed serially in some order..*
4. Durability: Once a transactions commits, its results are permanent.

Two-phase commit and its importance.

3 Reliability and Integrity

1. Consider the relation schemas:

Course

CrsCode	DeptId	CrsName	Descr

Transcript

StuId	CrsCode	Semester	Grade

Some constraints:

(a) All course codes must be unique in the Course relation.

Intra-relational. Key constraint. Name another “key.” Static.

Static constraints define legal instances.

(b) The course code in a transcript tuple must match a course code in a course tuple.

Inter-relational. Foreign key constraint. Static.

(c) A grade of “A” may not be changed to “I.”

Dynamic constraint.

Dynamic constraints define transitions between legal instances.

(d) A student may not take more than 21 credits per semester.

Semantic constraint. Implement business rules.