Relational Data Model I

Tom Kelliher, CS 318

Jan. 30, 2002

1 Administrivia

Announcements

Assignment

Read 4.3

From Last Time

PHP lab.

Outline

1. Schemas and abstraction.
2. Relational model.
3. Integrity constraints.

Coming Up

SQL data definition language I.

2 Schemas and Abstraction

1. Schema — description of the layout of data.

2. Database schemas:

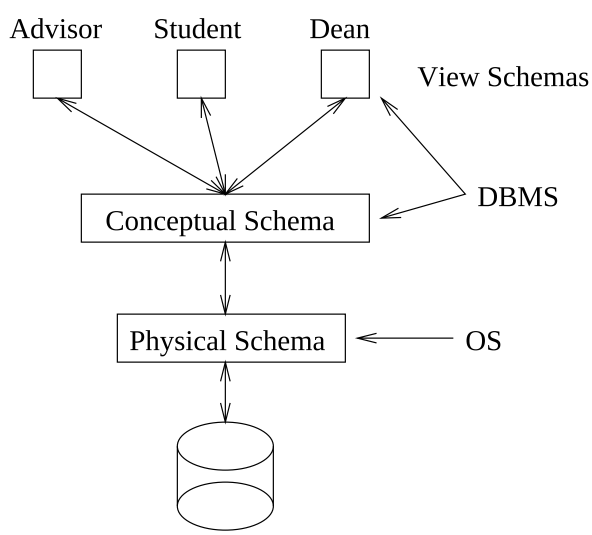
   (a) Physical: bits, bytes, and files level.

   (b) Conceptual: The database at the DBMS level.

   (c) View: Generally, constrictions of the conceptual schema. Supported by the DBMS.

3. Why abstractions?

4. Example:



   Views used to constrict the data available to a user. Enhance security, privacy, simplicity.

5. A **data model** consists of:

   (a) Conceptual and view schemas.

   (b) Constraints — conditions which the data must satisfy.

   (c) Data operations — queries, inserts, deletes, updates.

3 Relational Model

1. We've talked about **relation** before.

   Additionally, the **relational model** has well-defined, powerful mathematically-defined operators, enabling analysis and optimization of queries.

2. Schema (S) vs. instance (s).

3. Data atomicity — smallest unit of data the DBMS is aware of.

   (a) Example: The string type, even though strings decompose into chars.

   (b) What we're getting at: set-valued attributes such as "hobbies."

   (c) Solution: object-relational databases. (PostgreSQl.)

4. A **relation schema** consists of:

   (a) Relation name — unique across database.

   (b) Attributes and associated domains.

   (c) Integrity constraints — an instance must satisfy these to be legal.

   Type constraint example:

   i. Column naming: there must be a one-to-one mapping between columns in an instance and attributes in a schema.

   ii. Domain constraints: The values in a particular column of an instance must belong to the domain of the corresponding schema attribute.

5. **Relational database** — collection of relations.

   Database schema, database instance.

4 Integrity Constraints

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.

As opposed to structural constraints, as in some of the former constraints. (Which?)

4.1 Key Constraints

1. **Key constraint** definition:

`key`$(\overline{K})$ consists of a subset, $\overline{K}$, of attributes of S with the property that an instance, s, of S satisfies `key`$(\overline{K})$ if it does not contain a pair of distinct tuples whose values agree on all the attributes of $\overline{K}$. Also, we assume no proper subset of `key`$(\overline{K})$ is a key constraint.

2. What is the key for the Transcript table?

3. Superkeys.

4. A relation may have several keys, as we already saw.

    (a) Candidate keys: set of possible keys.

    Often, the candidate keys are expressed as ICs.

    (b) Primary key. Table may be indexed on this key.

4.2 Foreign Keys and Referential Integrity

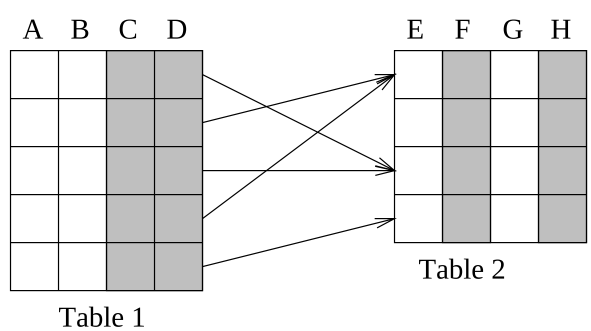
1. Tuples in one relation commonly reference tuples in another relation.

    Transcript references Course. How do you guarantee that a transcript row refers to an actual Course row?

2. **Referential integrity**: the referenced tuple must exist.

    Example of referential integrity: foreign key constraint.

3. Foreign key constraint:



    (a) (F, H) is a candidate key for Table 2.

(b) Foreign key (C, D) of Table 1 references the given candidate key of Table 2.

   1-1 relationship between the attributes and the corresponding attributes have the same values.

(c) Since the foreign key references a candidate key, at most one row of Table 2 is associated with a row of Table 1.

   For it to be a foreign key, "at most" must be "exactly."

4. Transcript(CrsCode) is a foreign key of Course: Transcript(CrsCode) references Course(CrsCode).

5. Formally:

   > Suppose that S and T are relation schemas, $\overline{F}$ is a list of attributes in S, and $\texttt{key}(\overline{K})$ is a key constraint in T. There is a 1-1 correspondence between attributes in $\overline{F}$ and $\overline{K}$. We say that relation instances $\mathbf{s}$ and $\mathbf{t}$ satisfy the foreign key constraint $S(\overline{F})$ references $T(\overline{K})$ and that $\overline{F}$ is a foreign key if and only if for every tuple $s \in \mathbf{s}$ there is a tuple $t \in \mathbf{t}$ that has the same values over the attributes of $\overline{K}$ as does s over the corresponding attributes of $\overline{F}$.

6. If the attribute set in Table 2 were not a candidate key, we would have an *inclusion dependency*.

   Foreign keys are a specific type of inclusion dependency.