The Relational Model
Constraints and Operations

CS157A

Chris Pollett

Sept. 21, 2005.

Outline

- Introduction
- Domain Constraints
- Key and Null Value Constraints
- Relational Databases and Schemas
- Entity Integrity, Referential Integrity, Foreign Keys
- Operations

Introduction

- The state of whole database depends on the state of each of its relations at a given time.
- We want to ensure this state satisfies certain constraints:
  - Constraint inherent in the data model. (**Inherent model based constraints**).
  - Constraints that are expressed by the schemas in the data model. (**schema based constraints**).
  - Constraints that must be enforced by the applications programs. (**application based constraints**).
- For this lecture will mainly focus on the second kind of constraints.

Domain Constraints

- These specify that each value of some attribute A must be an atomic value from dom(A).
- Some datatypes associated with domain constraints are:
  - numeric data types (short, integer, long)
  - real numbers (float, double)
  - characters of fixed and variable length
  - booleans
  - date, time, timestamps
  - money data types
  - sub-ranges of other data types, enumerated types

Key and Null Value Constraints

- Recall a relation r(R) is defined as a set of tuples.
- A subset SK of attributes of R is called a **superkey** if for any t1 != t2 in R, we have t1[SK] != t2[SK].
- A **key** K is a superkey such that no proper subset of K's attributes is also a superkey.
- For example, SSN is both a superkey and a key for EMPLOYEE.
- A typical R will have many keys. For example, First Name, Last Name might also be a key.
- Usually when we create the relation schema we will single one of these keys our and call it a **primary key**, all other keys will be called **candidate keys**.
- A key value typically cannot be a null value. We can specify this constraint in SQL using NOT NULL after the type.

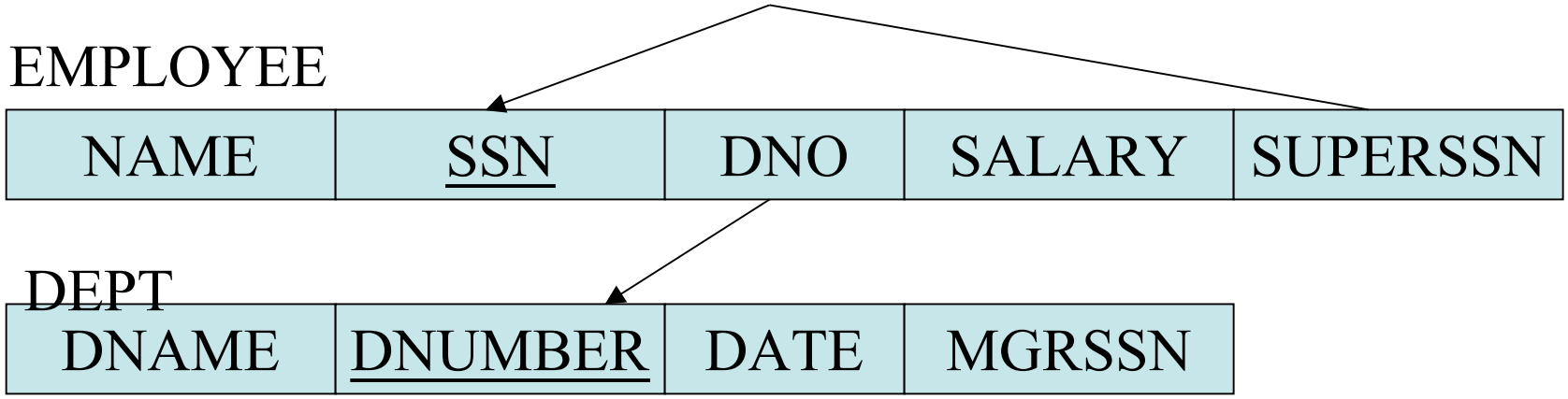
Relational Databases and Schemas

- So far we have only looked at single relations, so let's talk about a whole database.
- A set S={R1,R2…} of relation schemas is called a **relational database schema**.
- A set DB={r1(R1), r2(R2)..} of relations for each schema in S is called a **relational database state** or **relational database instance**.
- COMPANY={EMPLOYEE, DEPARTMENT, DEPT_LOCATIONS, PROJECT, WORK_ON, DEPENDENT} is an example relational database schema.
- A database state that so that each relation in it satisfies all of its integrity constraints is called **valid**; otherwise, the state is called **invalid**.

Entity Integrity, Referential Integrity, Foreign Keys

- The **entity integrity constraint** states that no primary key value can be null. It ensures that we can always find a tuple using its primary key.
- A **referential integrity constraint** is used to maintain consistency between the values in two or more relations. For instance the DNO of the EMPLOYEE table must be a DNumber in the DEPARTMENT table.
- To define this formally we say a set of attributes FK of R1 is a foreign key that references R2, if it satisfies:
  - the attributes of FK have the same domains as the primary key attributes PK of R2.
  - a value of FK in a tuple t1 of r1(R1) either occurs as value of PK for some t2 in r2(R2) or is null. The former case says t1[FK] =t2[PK].

Diagram of Foreign Key Constraint



Other Constraints

- Another kind of constraint that can occur in a database is a semantic constraint. For instance, the maximum number of hours an employee can work a week is 36.
- Such constraints can be enforced by the application program that updates the database. Or you might specify them with a general purpose **constraint specification language**.Trigger and assertions (CREATE ASSERTION in SQL-99) can be used.
- A functional dependency is another type of constraint. It might be written in the form X--> Y which says if we know the value of a tuple for X then its value for Y is uniquely determined.
- Finally, there are are transition constraints which can be used to control how the database changes from state to state. For example, my salary can only increase.

Operations

- There are two classes of operations typically done in the relational model: **retrievals** and **updates**.
- The **relational algebra** is used to define how to do retrievals.
- There are three common update operations: **Insert**, **Delete**, **Update**.

Insert

- An insert is used to add a tuple to a relation.
- It is accepted by the relation, if all the previously defined constraints are satisfied.
- An example insert might look like:

  Insert <‘bob’, ‘123456789’, ‘3’, ‘50000’, ‘345667789’> into EMPLOYEE.

DELETE

- A delete is used to remove a tuple from a table.
- Again, integrity constraints must be satisfied for it to apply.
- An example DELETE might look like:

Delete the WORKS_ON tuple with ESSN='123456789' and PNO=10;

Update

- An update operation is used to change the value of one or more attributes of a tuple.
- Again, integrity constraints must be satisfied.
- For example, Update the SALARY of the EMPLOYEE with SSN='123456789' to 60000.