

# Relational Databases - Tuples, Tables, Schemas, Relational Algebra

ECE 650  
Systems Programming & Engineering  
Duke University, Spring 2016

## Overview

- Relational model - Ted Codd of IBM Research in 1970
  - "A Relational Model of Data for Large Shared Data Banks"
- Attractive for databases
  - Simplicity + mathematical foundation
- Based on mathematical relations
  - Theoretical basis in set theory and first order predicate logic
- Implemented in a large number of commercial databases
  - E.g. ORACLE and Microsoft ACCESS

ECE 650 – Fall 2016

2

## Relational Model

- Represents database as a collection of *relations*
  - Think of a relation as a table of values
  - E.g.

	Name	Position	Department	Phone #
Employee Table	Reynolds	Manager	Sales	555-555-5444
	Smith	Engineer	Development	555-555-5555

- Relation as a table
  - Table name is called a **relation**
  - Each row represents a collection of related data values (**tuple**)
  - Columns help interpret meaning of values in each row; also called an **attribute**
    - All values in a column have the same *data type*
    - Data type of the values that can appear in column is called **domain**

ECE 650 – Fall 2016

3

## Domain

- What is a domain
  - Set of atomic values
    - Each value in domain is indivisible from relational model view
  - Commonly specified as a data type; often domain given a name
- Examples (logical definitions):
  - USA\_phone\_numbers: set of 10-digit phone #'s valid in US
  - Local\_phone\_numbers: set of 7-digit phone #'s value in area code
  - Names: Set of names of persons
  - Grade\_point\_averages: Set of real numbers between 0 and 4
- Name, data type, format:
  - USA\_phone\_numbers is char string of form (ddd)ddd-dddd
    - Where d is a decimal digit and first 3 digits are a valid area code

ECE 650 – Fall 2016

4

## Relation Schema

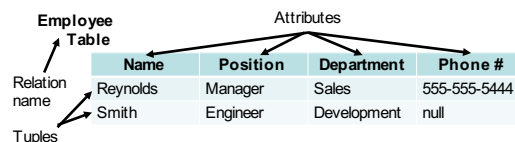
- Relation schema R denoted as  $R(A_1, A_2, \dots, A_n)$ 
  - Made up of relation name R and list of attributes  $A_1, A_2, \dots, A_n$
  - Attribute  $A_i$ 
    - Names a role played by some domain D in relation schema R
    - D is the domain of  $A_i$  and is denoted by  $\text{dom}(A_i)$
- Relation Schema describes a relation (named R)
- Degree of a relation is number of attributes n
- Example relation schema of degree 7:
  - STUDENT(Name, SSN, HomePhone, Address, OfficePhone, Age, GPA)

ECE 650 – Fall 2016

5

## Relation

- A **relation** of a relation schema R is denoted by  $r(R)$ 
  - Set of n-tuples:  $r = \{t_1, t_2, \dots, t_m\}$
  - Each n-tuple t is an ordered list of n values  $t = \langle v_1, v_2, \dots, v_n \rangle$ 
    - Where each value  $v_i$  is an element of  $\text{dom}(A_i)$  or NULL
    - The  $i$ th value in tuple t is referred to as  $t[A_i]$



ECE 650 – Fall 2016

6

## Relation (2)

- Stated another way
  - Relation  $r(R)$  is a mathematical relation of degree  $n$  on the domains  $\text{dom}(A_1), \text{dom}(A_2), \dots, \text{dom}(A_n)$
  - Which is a subset of the Cartesian product of the domains of  $R$ 
    - $r(R) \subseteq (\text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n))$
    - Cartesian product specifies all possible combinations
    - Cardinality of domain  $D$  is  $|D|$ ; # of tuples in Cartesian product is:
      - $|\text{dom}(A_1)| * |\text{dom}(A_2)| * \dots * |\text{dom}(A_n)|$
  - Current relation state:
    - Reflects only valid tuples that represent particular state of real world
    - Schemas are relatively static (change very infrequently)
    - But current relation state may change frequently
  - Possible for several attributes to have the same domain
    - But attributes indicate different roles of the domain
      - Eg. HomePhone vs. OfficePhone

## Relational Model Notation

- Relation schema  $R$  of degree  $n$  is denoted by  $R(A_1, A_2, \dots, A_n)$
- $N$ -tuple  $t$  in a relation  $r(R)$  is denoted by  $t = \langle v_1, v_2, \dots, v_n \rangle$ 
  - $v_i$  is the value corresponding to attribute  $A_i$
  - $t[A_i]$  refers to the value  $v_i$  in  $t$  for Attribute  $A_i$
- Letters  $Q, R, S$  denote relation names
- Letters  $q, r, s$  denote relation states
- Letters  $t, u, v$  denote tuples
- $R.A$  denotes the relation name to which an attribute belongs
  - Since the same name may be used for attributes in different relations

## Definition Summary

Informal Terms	Formal Terms
Table	Relation
Column Header	Attribute
All Possible Column Values	Domain
Row	Tuple
Table Definition	Schema of a Relation
Populated Table	State of the Relation

## Relational Constraints

- Restrictions on data that can be specified on a relational database schema
- Domain Constraints
- Key Constraints
- Constraints on NULL
- Entity Integrity Constraint
- Referential Integrity Constraint

## Domain Constraints

- Value of each attribute  $A$  must be atomic value from  $\text{dom}(A)$
- Data types include standard numeric types
  - Integer, long integer
  - Float, double-precision float
- Also characters, fixed-length and variable-length strings
- Others
  - Date, timestamp, money data types
  - Enumerated data types
- Will discuss more when we talk about SQL

## Key Constraints

- All tuples in a relation must be distinct
  - No 2 tuples can have same values for all attributes
- Superkey
  - Set of attributes where no 2 tuples can have the same values
  - Every relation has at least one default superkey (all attributes)
- Key
  - Superkey with property that removing any attribute from the set leaves a set that is not a superkey of the relation schema
- Example
  - $\text{STUDENT}(\text{Name}, \text{SSN}, \text{HomePhone}, \text{Address}, \text{OfficePhone}, \text{Age}, \text{GPA})$
  - Attribute set  $\{\text{SSN}\}$  is key (no 2 students can have same value)
  - Attribute set  $\{\text{SSN}, \text{Name}, \text{Age}\}$  is a superkey (but not a key)

## Key Constraints & Entity Integrity

- Value of key attribute uniquely identifies each tuple
- Set of attributes constituting a key is a property of the relation schema
  - Should hold on "every" relation state of the schema
  - Time-invariant: should hold even as tuples are added
- A relation schema may have more than one key
  - Each is called a candidate key; one is designated as **primary key**
  - Convention to underline the primary key of a relation schema

CAR LicenseNum EngineSerialNum Make Model Year

- Entity Integrity Constraint
  - Primary key value cannot be NULL

## NULL Constraints

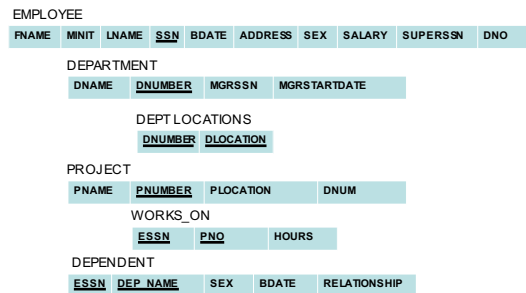
- NULL may or may not be permitted for different attributes
- E.g. if Name attribute must have a valid, non-null value
  - It is said to be constrained to be NOT NULL

## Relational Database

- Contains many relations
- Tuples in relations are related in various ways
- Relational database schema
  - Set of relation schemas  $S = \{R_1, R_2, \dots, R_m\}$
  - Set of integrity constraints (IC)

## Example Relational Database Schema

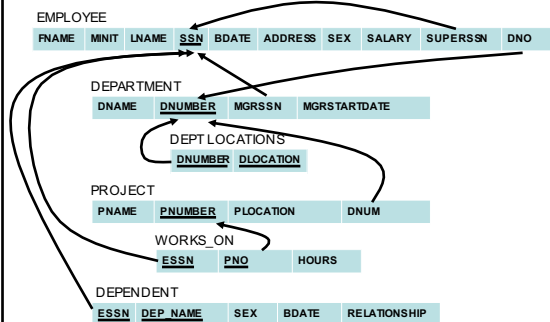
- COMPANY = {EMPLOYEE, DEPARTMENT, DEPT\_LOCATIONS, PROJECT, WORKS\_ON, DEPENDENT}



## Referential Integrity Constraint

- Specified between 2 relations
- Maintains consistency among tuples of two relations
- Informally
  - Tuple in a relation that refers to another relation must refer to an existing tuple in that relation
- Formally
  - For ref integrity constraint between  $R_1$  &  $R_2$ , define *foreign key*
  - Set of attributes FK in  $R_1$  is foreign key referencing  $R_2$  if:
    - Attributes in FK have same domain(s) as the primary key attributes PK of  $R_2$  (attributes FK thus refer to the relation  $R_2$ )
    - A value of FK in tuple  $t_1$  of current state  $r_1(R_1)$  either occurs as a value of PK for some tuple  $t_2$  in  $r_2(R_2)$  or is NULL

## Example Referential Integrity Constraints



## Other Constraints

- Semantic Integrity Constraints
  - E.g. salary of employee should not exceed salary of supervisor
  - E.g. max hours an employee can work on all projects per week
  - Can be specified via a constraint specification language
    - Via mechanisms called triggers or assertions
- Transition Constraints
  - Deal with state changes in the database
  - E.g. tenure length of an employee can only increase
  - Specified using rules and triggers

## Relational Model Operations

- Updates
  - Insert, delete, modify
  - Integrity constraints must not be violated
- Retrievals
  - Involve relational algebra operations

## Insert

- Provides list of attribute values for new tuple t to be inserted into relation R
- Can possibly violate several constraints
  - Domain: attribute value doesn't appear in corresponding domain
  - Key: key value in new tuple t already exists in another tuple
  - Entity: primary key of new tuple t is NULL
  - Referential: foreign key in t refers to a tuple that does not exist
- Example (see example COMPANY database)
  - Insert <'Cecilia', 'F', 'Kolonsky', null, '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, null, 4> into EMPLOYEE
    - Entity integrity constraint violation; insert is rejected

## Delete

- Can only violate referential integrity
  - If tuple being deleted is referenced by foreign keys in other tuples
- Specify a deletion
  - Give a condition on the attributes of the tuple(s) of a relation
  - E.g. delete tuple with attributes matching given values
- Options if a deletion causes a violation
  - Reject the deletion operation
  - Cascade the deletion
    - Delete tuples that reference the tuple being deleted
  - Modify the referencing attribute values
    - E.g. change them to NULL

## Update

- Change values of attribute(s) in tuple(s) of a relation
- Specify a condition on the attributes of the relation to select tuple(s) to be modified
- E.g. update SALARY of EMPLOYEE tuple with SSN='999887777' TO 28000
- Usually always valid to update attribute that is not a key
  - Either primary or foreign
  - Check is done to make sure new value is correct type and domain
- Modifying a primary key: like delete + insert
- Modifying a foreign key: check referential integrity

## Relational Algebra Operations

- Data models must include a set of ops to manipulate data
- Relational Algebra
  - Basic set of relational model operations
- Ops allow users to specify basic data retrieval requests
  - Result of retrieval is a new relation
    - May have been formed from one or more other relations
  - Result relations can be further manipulated with further ops
- Sequence of relational algebra ops form an "expression"
- Relational algebra operations:
  - Set ops: union, intersection, set difference, Cartesian product
  - Ops specifically for relational databases: select, project, join

## SELECT Operation

- Essentially a filter over a relation
  - Forms a new relation with only tuples matching a condition
  - Resulting relation has same degree & attributes as original relation
- $\sigma_{\langle \text{selection condition} \rangle}(R)$ 
  - E.g.  $\sigma_{(DNO=4 \text{ AND } SALARY > 50000)}(EMPLOYEE)$
  - R is a relation
    - Could be a database relation or result of another select
  - Selection condition can compare ( $=$ ,  $<$ ,  $<=$ ,  $>$ ,  $>=$ ,  $\neq$ )
  - Selection condition clauses can be combined (AND, OR, NOT)
- SELECT operation applies independently to each tuple
  - Resulting number of tuples is less than or equal to original relation
- Note that SELECT is commutative
  - Chain of SELECT ops can be applied in any order

## PROJECT Operation

- PROJECT chooses certain columns of a relation
  - Recall SELECT chooses certain rows of a relation
  - Other columns are discarded
- $\pi_{\langle \text{attribute list} \rangle}(R)$ 
  - E.g.  $\pi_{(LNAME, FNAME, SALARY)}(EMPLOYEE)$
  - Result has only attributes shown in list (in same order as listed)
  - If list only includes non-key attributes, there may be duplicates
    - Duplicate tuples are removed by PROJECT operation
- Commutativity does not hold for PROJECT operation

## Sequences of Operations & RENAME

- If we want to apply several ops one after the other
  - Can either write as a single expression (via nesting)
  - Or can apply one op at a time and save intermediate relations
- Example:
  - get first name, last name, salary of all employees in dept 5
  - $\pi_{(LNAME, FNAME, SALARY)}(\sigma_{DNO=5}(EMPLOYEE))$  \*or\*
  - $DEP5\_EMPS = \sigma_{DNO=5}(EMPLOYEE)$
  - $RESULT = \pi_{(LNAME, FNAME, SALARY)}(DEP5\_EMPS)$
- Can also use to rename attributes
  - Sometimes useful for UNION and JOIN as we'll see
  - $R(FIRSTNM, LASTNM, SALARY) = \pi_{(LNAME, FNAME, SALARY)}(TMP)$

## Set Theoretic Ops

- UNION, INTERSECTION, SET DIFFERENCE
  - $\cup, \cap, -$
- Binary ops applied to two sets
- Relations must be *union compatible*
  - Have same degree n, and  $\text{dom}(A_i) = \text{dom}(B_i)$  for all  $1 \leq i \leq n$
- Example:
  - Find SSN of all employees who work in dept 5 or supervise an employee in dept 5
  - $DEP5\_EMPS = \sigma_{DNO=5}(EMPLOYEE)$
  - $RESULT1 = \pi_{SSN}(DEP5\_EMPS)$
  - $RESULT2(SSN) = \pi_{SUPERSSN}(DEP5\_EMPS)$
  - $RESULT = RESULT1 \cup RESULT2$

## Cartesian Product

- Also called cross product or cross join (denoted by  $\times$ )
- Combines tuples from 2 relations
  - Resulting relation has attributes of both original relations
- Commonly used followed by a SELECT
  - That matches attributes coming from both component relations
- Example:
  - For each female employee get a list of names of her dependents
  - $FEMALE\_EMPS = \sigma_{SEX=F}(EMPLOYEE)$
  - $EMP\_NAMES = \pi_{(FNAME, LNAME, SSN)}(FEMALE\_EMPS)$
  - $EMP\_DEPENDENTS = EMP\_NAMES \times DEPENDENT$
  - $ACTUAL\_DEPENDENTS = \sigma_{SSN=ESSN}(EMP\_DEPENDENTS)$
  - $RESULT = \pi_{(FNAME, LNAME, DEPENDENT\_NAME)}(ACTUAL\_DEPENDENTS)$

## JOIN Operation

- Useful to combined related tuples (denoted by  $\bowtie$ )
- Example:
  - Retrieve name of manager of each department
  - $DEPT\_MGR = DEPARTMENT \bowtie_{MGRSSN=SSN} EMPLOYEE$
  - $RESULT = \pi_{(DNAME, LNAME, FNAME)}(DEPT\_MGR)$
- Essentially does a Cartesian Product, then SELECT
  - General condition is:  $\langle \text{cond} \rangle \text{ AND } \langle \text{cond} \rangle \text{ AND } \dots \text{ AND } \langle \text{cond} \rangle$
  - Where cond is of form  $A_i \theta B_j$  (called THETA JOIN)
    - $A_i$  and  $B_j$  are attributes of R and S
  - If cond is testing equality, then called EQUIJOIN
    - Note always will have a pair of attributes with equal values
  - NATURAL JOIN (denoted  $\Join$ ) automatically gets rid of second (superfluous) attribute