Database Technology

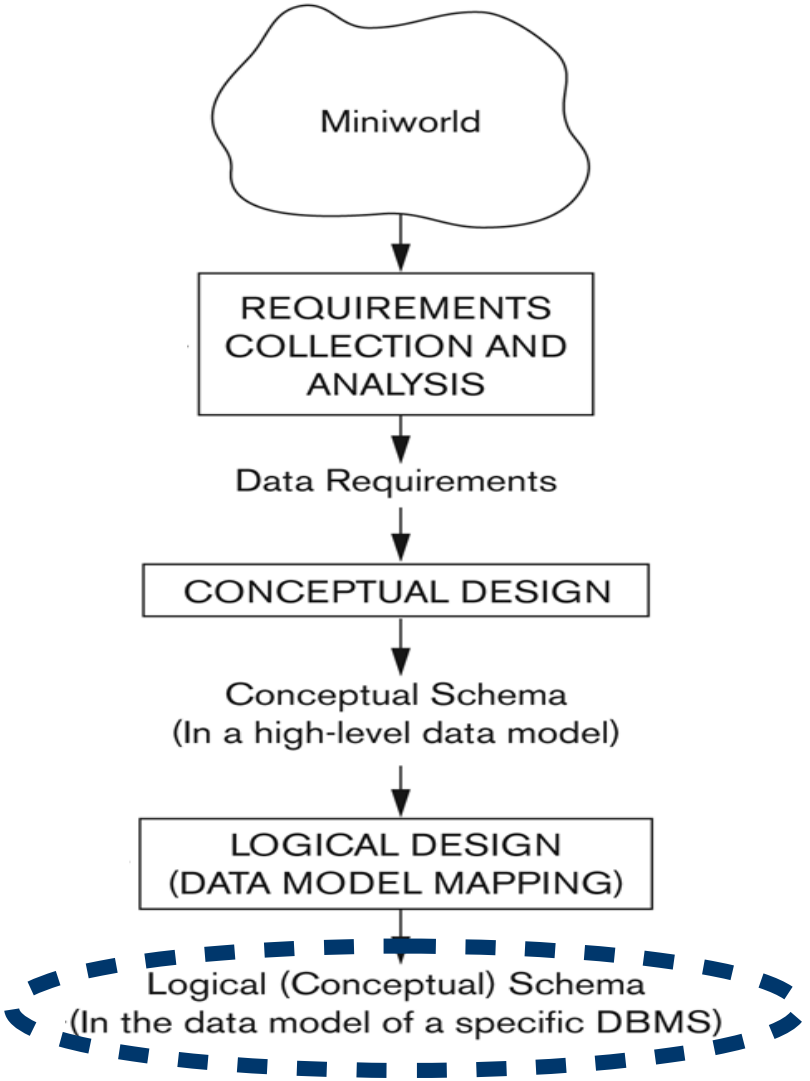
Topic 2: Relational Databases

Olaf Hartig

olaf.hartig@liu.se

LINKÖPING UNIVERSITY

Recall: DB Design Process



Relational Data Model

Relational Model Concepts

- Relational database: represent data as a collection of *relations*
  - Think of a relation as a table of values

Relation Name: **STUDENT**

Attributes

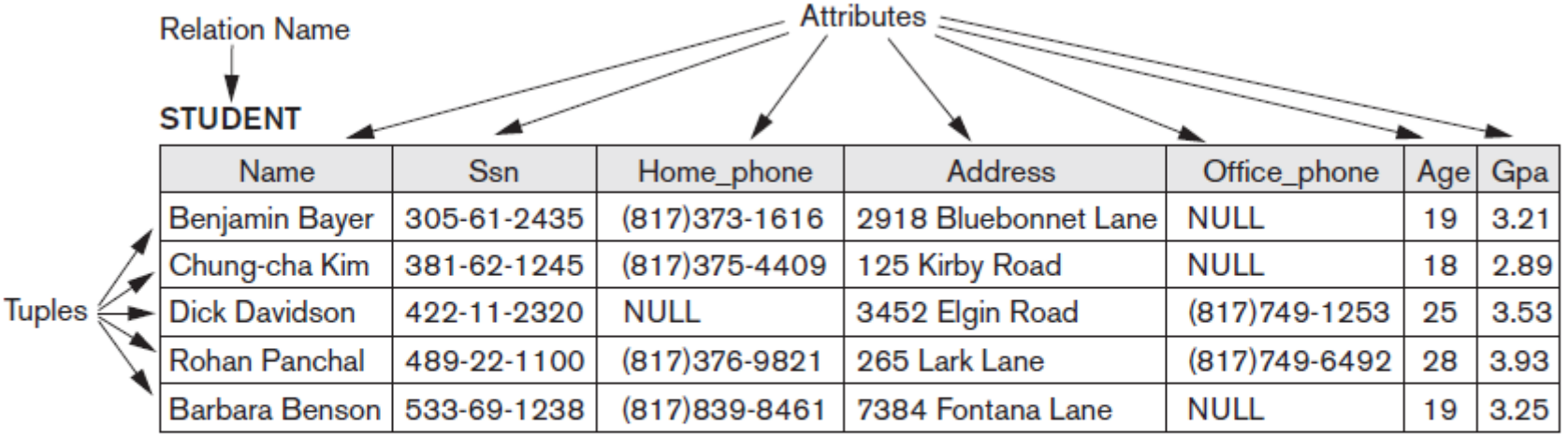
Tuples

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

- Each row (*tuple*) represents a record of related data values
  - Facts that typically correspond to a real-world entity or relationship
- Each column (*attribute*) holds a corresponding value for each row
  - Columns associated with a data type (*domain*)
  - Each column header: *attribute name*

Relational Model Concepts (cont'd)

- Relational database: represent data as a collection of *relations*
  - Think of a relation as a table of values



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

- **Schema** describes the relation
  - Relation name, attribute names and domains
  - Integrity constraints
- **Instance** (also called **state**) denotes the *current* contents of the relation
  - *Set* of tuples

Domains

- **Domain** is a set of *atomic* values
  - { 0, 1, 2, … }
  - { Jo Smith, Dana Jones, Ashley Wong, Y. K. Lee, … }

- **Atomic**: Each value indivisible

- Domains specified by **data type** rather than by enumeration
  - Integer, string, date, real, etc.
  - Can be specified by format
    - e.g., *(ddd)ddd-dddd* for phone numbers
      (where *d* represents a digit)

Schemas and Attributes

- **Relation schema**
  - A relation name *R* and a list of attributes *A1*, *A2*, ... , *An*
  - Denoted by *R*(*A1*, *A2*, ..., *An*)

- **Attribute** *Ai*
  - Name of a role in the relation schema *R*
  - Associated with a domain **dom(*Ai*)**
  - Attribute names do not repeat within a relation schema, but domains can repeat

- **Degree** (or **arity**) of a relation
  - Number of attributes *n* in its relation schema

NULL Values

- Each domain may be augmented with a special value called NULL
  - Represent the values of attributes that may be unknown or may not apply to a tuple
  - If an attribute of a tuple is NULL, we cannot make any assumption about the value for that attribute (for that tuple)
- Interpretations for NULL values
  - Nothing is known about the value
  - Value exists but is (currently) not available
  - Value undefined (i.e., attribute does not apply to this tuple)
- For instance, Ashley's telephone number is NULL could mean
  - Ashley doesn't have a phone
  - Ashley has a phone but we don't know the number (perhaps withheld)
  - Ashley has a phone that has no number

Integrity Constraints

What are Integrity Constraints?

- Restrictions on the permitted values in a database instance / state
  - Derived from the rules in the miniworld that the DB represents

1. **Inherent model-based constraints** (also called **implicit constraints**)
   - Inherent in the data model, enforced by DBMS
   - e.g., duplicate tuples are not allowed in a relation
2. **Schema-based constraints** (also called **explicit constraints**)
   - Can be expressed in schemas of the data model, enforced by DBMS
   - e.g., films have only one director
   - Our focus here
3. **Application-based** (also **semantic constraints** or **business rules**)
   - Not directly expressed in schemas
   - Expressed and enforced by application program
   - e.g., this year's salary increase can be no more than last year's

Uniqueness Constraints

- Let *R* be a relation and *K* be a (sub)set of attributes of *R*
- If we specify the uniqueness constraint for *K*, then for any pair of tuples in *R*, the tuples must have a different value for at least one of the attributes in *K*

- Uniqueness must hold in all valid instances of *R*
- Uniqueness serves as a constraint on updates

Student

PN	FName	LName
19970218-1782	Jennifer	Li
19951223-6512	Paul	Smith
19990721-1222	Kim	Jonsson

Grade

Course	StPN	Grade
TDDD17	19970218-1782	4
TDDD43	19970218-1782	5
TDDD43	19951223-6512	3

Superkeys and Candidate Keys

- A set *K* of attributes of *R* is called a *superkey* of R if it has the

  Uniqueness property: no two distinct tuples have the same values across all attributes in *K* (i.e., we may define a uniqueness constraint for *K*)

- *K* is called a *key* of *R* if, additionally, it **also** has the

  Minimality property: no proper subset of *K* has the uniqueness property

- Hence, every key is a superkey, but not every superkey is a key

- "candidate key" = key
  - used, in particular, if multiple different keys are possible

Primary Key

- There may be *more than one* candidate key in a relation
- **Primary key**: a particular candidate key is *chosen* as the primary
  - Diagrammatically, underline its attribute(s)
  - Tuples cannot have NULL for any primary key attribute
- Other candidate keys are designated as **unique**
  - Non-NULL values cannot repeat, but values may be NULL

Person1

PN	Name
19970218-1782	Jennifer
19970218-1782	Paul
19990721-1222	Jennifer

Person2

PN	Name
19970218-1782	Jennifer
19970218-1782	Paul
19990721-1222	Jennifer

Other Schema-Based Integrity Constraints

- **Entity integrity constraint**: No primary key value can be NULL
- **Domain constraint**: declared by specifying the datatype (domain) of the attributes
- **Referential integrity constraint**
  - see next slides

Referential Integrity Constraints (Motivation)

- Consider the following two relations

Student

PN	Name
19970218-1782	Jennifer
19951223-6512	Paul
19990721-1222	Kim

Grade

Course	StPN	Grade
TDDD17	19970218-1782	4
TDDD43	19970218-1782	5
TDDD43	19951223-6512	3

- We may want to make sure that for every student for which we record grades (in the Grade relation) we have a record in the Student relation
- That is, assuming the given instance of the Student relation, it would be invalid to have the following tuple in the Grade relation:

  (TDDD17, 20010219-6678, 4)

Referential Integrity Constraints

- Maintains consistency among tuples in two relations
- Allows every tuple in one relation to refer to a tuple in another
- Formally:
  - Let *PK* be the primary key in a relation *R1*
    - e.g., *PK* = { PN } in the Student relation on the previous slide
  - Let *FK* be a set of attributes for another relation *R2*
    - e.g., *FK* = { StPN } in the Grade relation on the previous slide
  - The attribute(s) *FK* have the same domain(s) as the attribute(s) *PK*
  - Constraint: For every tuple *t2* in *R2*, either

    i) there is a tuple *t1* in *R1* such that the value that *t1* has for *PK* is the same as the value that *t2* has for *FK*, or

    ii) the value that *t2* has for *FK* is NULL
    - e.g., for every tuple *t2* in the Grade relation, there is a tuple *t1* in the Student relation such that the PN value of *t1* is the same as the StPN value of *t2*, or the StPN value of *t2* is NULL

Diagramming Referential Constraints

- Show each relational schema
  - Underline primary key attributes in each
- Directed arc from each foreign key to the relation it references

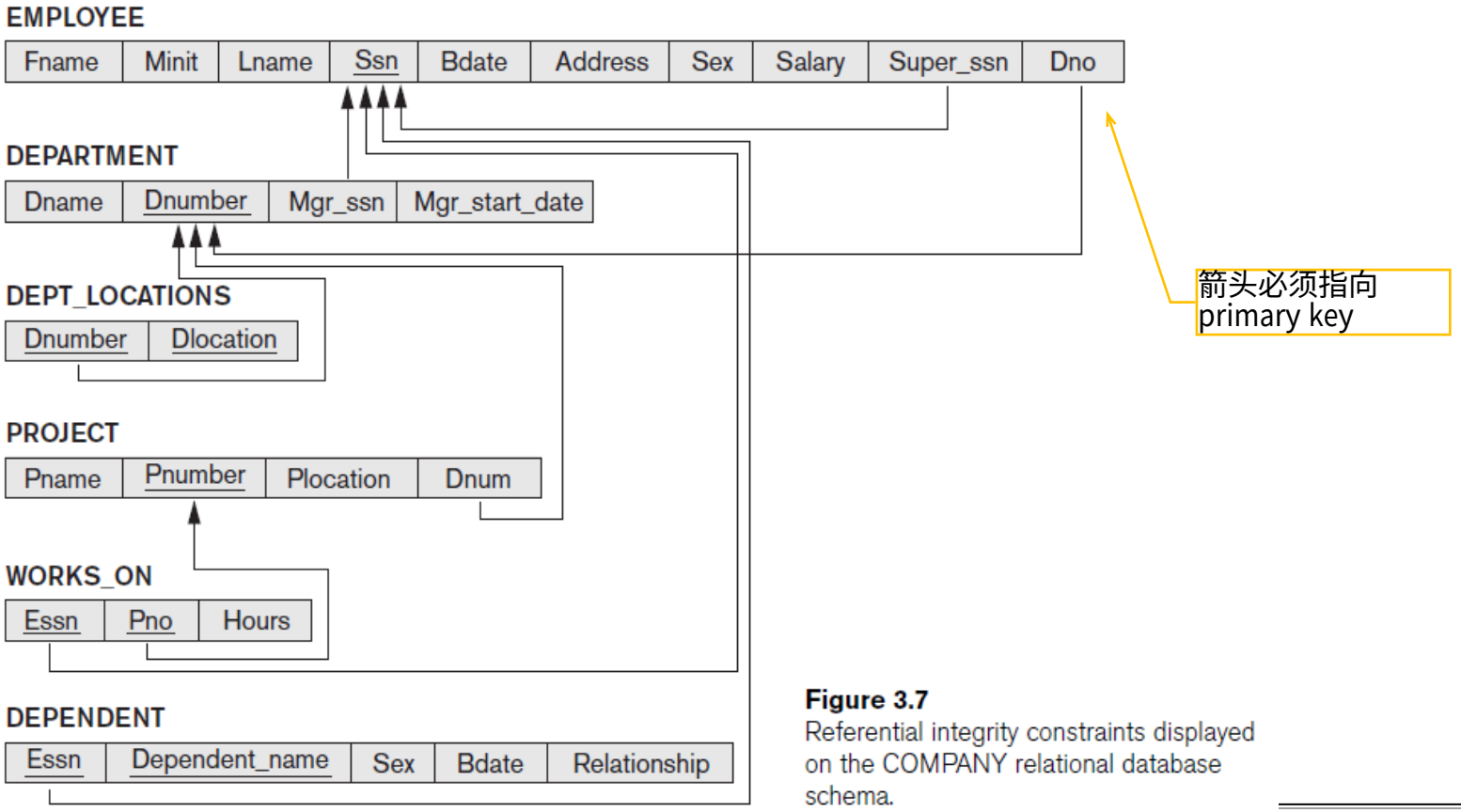


Figure 3.7
Referential integrity constraints displayed on the COMPANY relational database schema.

www.liu.se

LINKÖPING UNIVERSITY