Database Technology

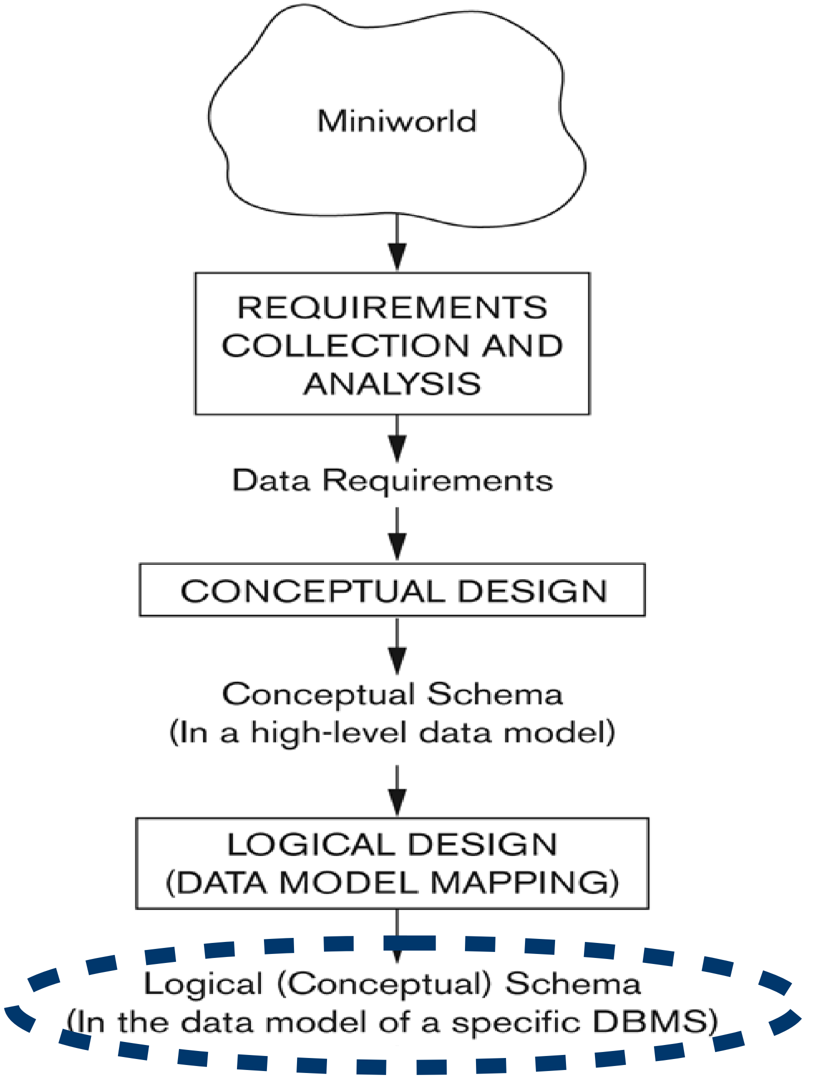
Topic 2: Relational Databases

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*
- Example relation:

Relation Name

Attributes

STUDENT

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

- **Quiz:** each of these things is called a …

1. record / 2. tuple / 3. row

… in the relation data model.

Relational Model Concepts (cont'd)

- Relational database: represent data as a collection of *relations*
- Example relation:

Relation Name

Attributes

Tuples

STUDENT

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 and consists of:
  - Relation name
  - Attributes, each of which has a name and a domain
  - Integrity constraints
- **Instance** (also called **state**) is the *current* content 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)

Quiz (NULL Values)

- Notice the value NULL that the Barbara Benson tuple has for the Office_phone attribute

Relation Name

Attributes

Tuples

STUDENT

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

- What can this value mean?

  1) Barbara Benson doesn’t have an office phone.

  2) Barbara Benson has an office phone but we don’t know the number (perhaps withheld).

  3) Any of the previous two.

Quiz

- A relation schema consists of:

  A) relation name, attribute names and domains, and tuples;

  or

  B) relation name, attribute names and domains, and restrictions;

  or

  C) relation name, tuples, and NULL values.

Quiz

- A relation schema consists of:

  A) relation name, attribute names and domains, and tuples;

  or

  B) relation name, attribute names and domains, and ~~restrictions;~~ integrity constraints

  or

  C) relation name, tuples, and NULL values.

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

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

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

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

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*
- *K* is called a *superkey*
- If *K* is minimal (no redundant attributes), it is called a *key*
  - hence, every key is a superkey, but not every superkey is a key

Group Activity

- 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*
- *K* is called a *superkey*
- If *K* is minimal (no redundant attributes), it is called a *key*
  - hence, every key is a superkey, but not every superkey is a key

- For the CAR relation,
  - specify a key, and
  - specify 2 superkeys that are *not* a keys

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

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

<u>PN</u>	Name
19970218-1782	Jennifer
19951223-6512	Paul
19990721-1222	Kim

Grade

<u>Course</u>	<u>StPN</u>	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

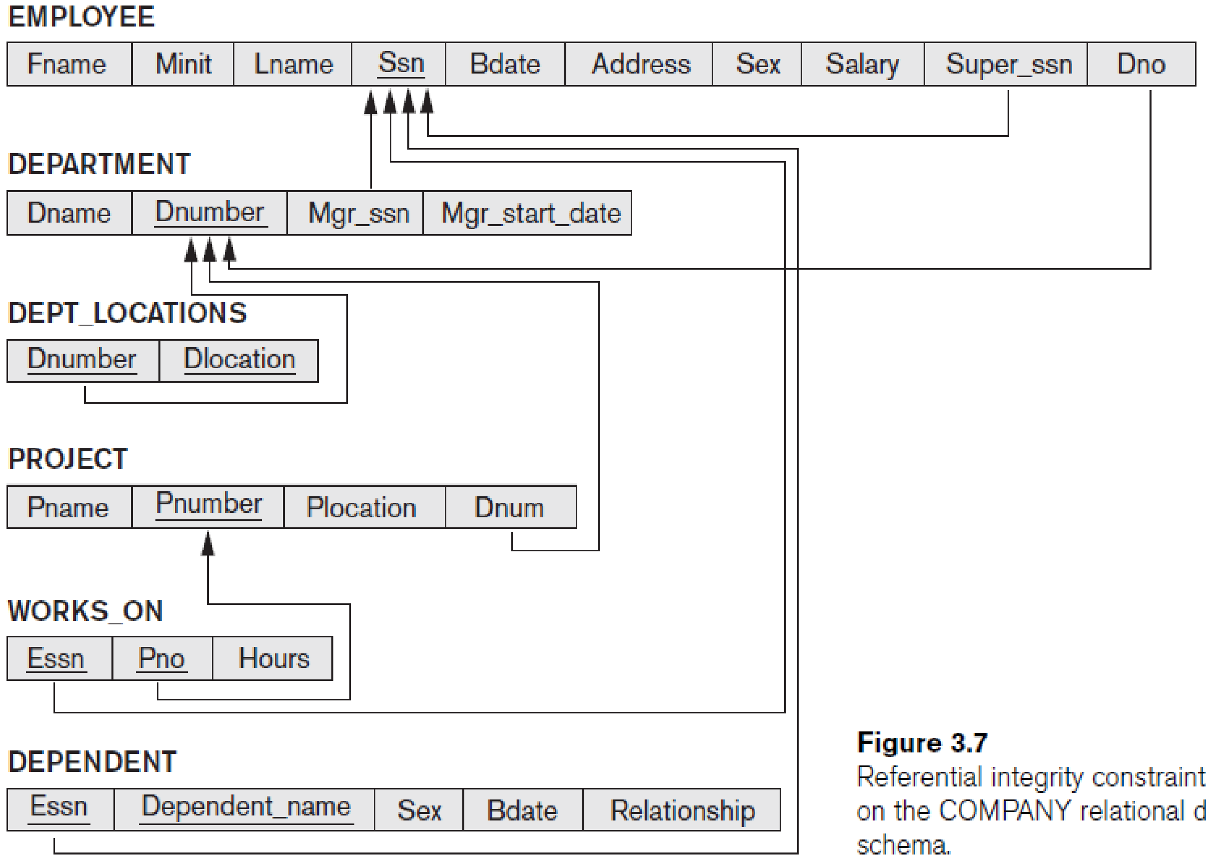


Figure 3.7
Referential integrity constraints displayed on the COMPANY relational database schema.

Quiz

- Consider the following two relations

Instructor

<u>**ID**</u>	**Name**	**Office**
4	Jennifer	B308
35	Paul	B311
12	Kim	E112

Course

<u>**CourseID**</u>	<u>**Year**</u>	**Instructor**
cid444	2012	35
cid598	2013	4
cid444	2013	35

- Which of the following statements is correct?
  - (a) We can insert a new *Course* tuple (cid598,2017,2).
  - (b) We can modify the two cid444 *Course* tuples by changing their *Instructor* value to 12.
  - (c) We can modify the cid598 *Course* tuple by changing its *CourseID* value to cid444.

www.liu.se

LINKÖPING UNIVERSITY