ENSF 608: Understanding and Mapping the Relational Data Model

Dr. Emily Marasco

Fall 2021

Textbook: Fundamentals of Database Systems, 7th Ed., Elmasri & Navathe

Copyright © 2020 Marasco, 2016 Pearson Education, Inc.

Constraints

Constraints determine which values are permissible and which are not in the database.

They are of three main types:

1. **Inherent or Implicit Constraints:** These are based on the data model itself. (E.g., relational model does not allow a list as a value for any attribute)
2. **Schema-based or Explicit Constraints:** They are expressed in the schema by using the facilities provided by the model. (E.g., max. cardinality ratio constraint in the ER model)
3. **Application based or semantic constraints:** These are beyond the expressive power of the model and must be specified and enforced by the application programs.

Copyright © 2020 Marasco, 2016 Pearson Education, Inc.

Relational Integrity Constraints

- Constraints are **conditions** that must hold on **all** valid relation states.
- There are three **main types** of (explicit schema-based) constraints that can be expressed in the relational model:
  - **Key** constraints
  - **Entity integrity** constraints
  - **Referential integrity** constraints
- Another schema-based constraint is the **domain** constraint
  - Every value in a tuple must be from the **domain of its attribute** (or it could be **null**, if allowed for that attribute)

Copyright © 2020 Marasco, 2016 Pearson Education, Inc.

Key Constraints (1 of 3)

- **Superkey** of *R*:
  - Is a set of attributes SK of *R* with the following condition:
    - No two tuples in any valid relation state *r*(*R*) will have the same value for SK
    - That is, for any distinct tuples t_1 and t_2 in $r(R)$, $t_1[\text{SK}] \neq t_2[\text{SK}]$
    - This condition must hold in **any valid state** *r*(*R*)
- **Key** of *R*:
  - A "minimal" superkey
  - That is, a key is a superkey *K* such that removal of any attribute from *K* results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)
- A Key is a Superkey but not vice versa

Copyright © 2020 Marasco, 2016 Pearson Education, Inc.

Key Constraints (2 of 3)

- Example: Consider the CAR relation schema:
  - CAR (State, Reg#, SerialNo, Make, Model, Year)
  - CAR has two keys:
    - Key1 = {State, Reg#}
    - Key2 = {SerialNo}
  - Both are also superkeys of CAR
  - {Serial No, Make} is a superkey but **not** a key.
- In general:
  - Any **key** is a **superkey** (but not vice versa)
  - Any set of attributes that **includes a key** is a **superkey**
  - A **minimal** superkey is also a key

Copyright © 2020 Marasco, 2016 Pearson Education, Inc.

Key Constraints (3 of 3)

- If a relation has several **candidate keys,** one is chosen arbitrarily to be the **primary key.**
  - The primary key attributes are **underlined.**
- Example: Consider the CAR relation schema:
  - CAR (State, Reg#, SerialNo, Make, Model, Year)
  - We chose SerialNo as the primary key
- The primary key value is used to **uniquely identify** each tuple in a relation
  - Provides the tuple identity
- Also used to **reference** the tuple from another tuple
  - General rule: Choose as primary key the smallest of the candidate keys (in terms of size)
  - Not always applicable – choice is sometimes subjective

Copyright © 2020 Marasco, 2016 Pearson Education, Inc.

Car Table with Two Candidate Keys – LicenseNumber Chosen as Primary Key

Figure 5.4 The CAR relation, with two candidate keys: License_number and Engine_serial_number.

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

Copyright © 2020 Marasco, 2016 Pearson Education, Inc.