# Relational Database Constraints

Chapter 3 b

# Chapter Outline

- Relational integrity constraints
  - Key constraints
  - Entity integrity
  - Referential integrity
  - Other types of constraints
- Example
- ER to Relational mapping algorithm

# Relational Integrity Constraints

- Constraints are *conditions* that must hold on *all* valid relation instances. There are three main types of constraints:
  1. **Key** constraints
  2. **Entity integrity** constraints
  3. **Referential integrity** constraints

# Key Constraints

- **Superkey** of R: A set of attributes SK of R such that no two tuples *in any valid relation instance r(R)* will have the same value for SK. That is, for any distinct tuples t1 and t2 in r(R), t1[SK] ≠ t2[SK].
- **Key** of R: A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

# Key Constraints II

**Example**: The CAR relation schema:

CAR(Label, Reg#, SerialNo, Make, Model, Year)

has two keys Key1 = {Label, Reg#}, Key2 = {SerialNo}, which are also superkeys. {SerialNo, Make} is a superkey but *not* a key.

- If a relation has *several* **candidate keys**, one is chosen arbitrarily to be the **primary key**. The primary key attributes are *underlined*.

# Entity Integrity

- **Relational Database Schema**: A set S of relation schemas that belong to the same database. S is the *name* of the **database**. $S = \{R_1, R_2, ..., R_n\}$
- **Entity Integrity**: The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of r(R). This is because primary key values are used to *identify* the individual tuples.

  $t[PK] \neq null$ for any tuple t in r(R)

# Referential Integrity

- A constraint involving *two* relations (the previous constraints involve a *single* relation).
- Used to specify a *relationship* among tuples in two relations: the **referencing relation** and the **referenced relation**.
- Tuples in the *referencing relation* $R_1$ have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the *referenced relation* $R_2$. A tuple $t_1$ in $R_1$ is said to **reference** a tuple $t_2$ in $R_2$ if $t_1[FK] = t_2[PK]$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from $R_1$.FK to $R_2$.

# Referential Integrity II

Statement of the constraint

The value in the foreign key column(s) FK of the **referencing relation** $R_1$ can be either:

1. a value of an existing primary key value of the corresponding primary key PK in the **referenced relation** $R_2$, or.
2. a null.

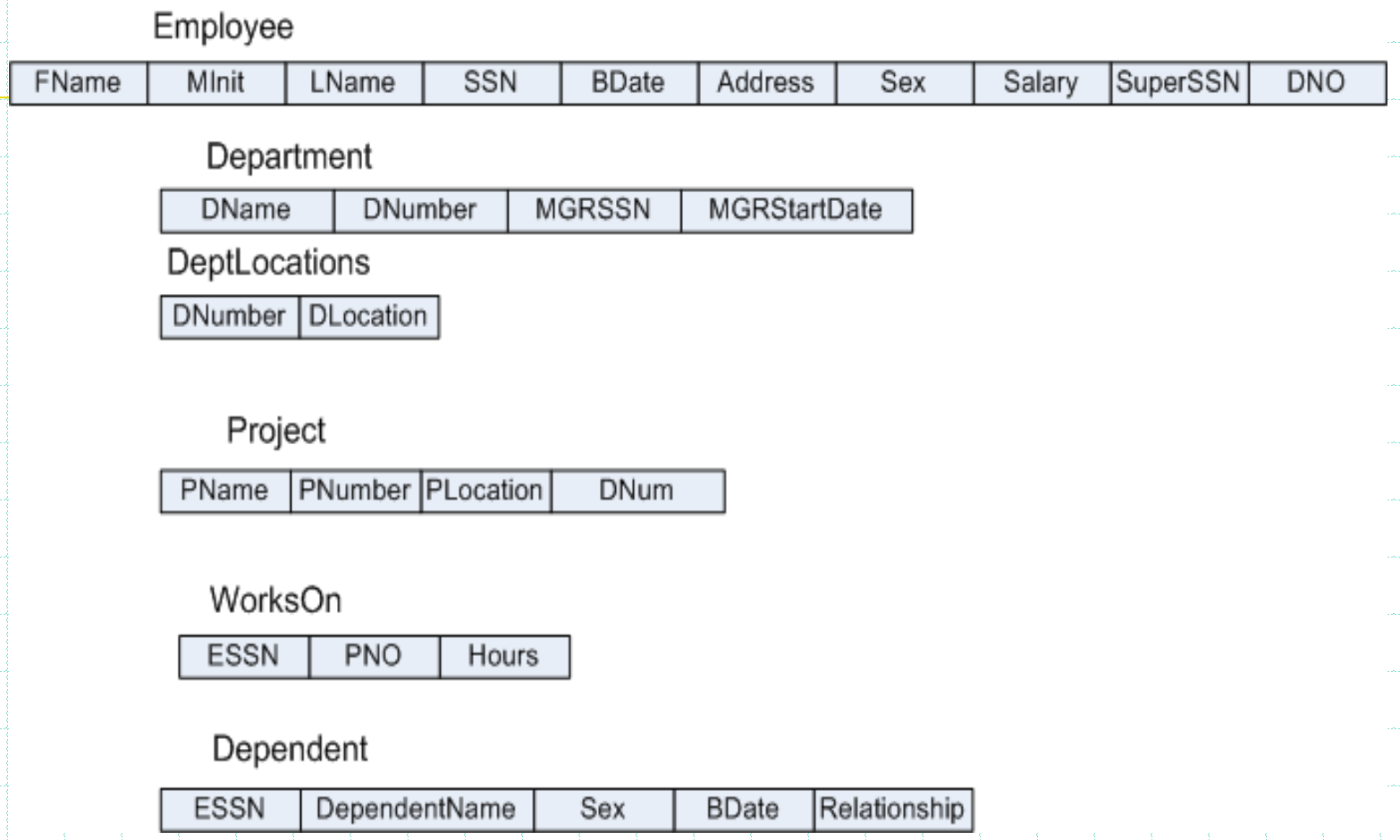
In case (2), the FK in $R_1$ should not be a part of its own primary key.

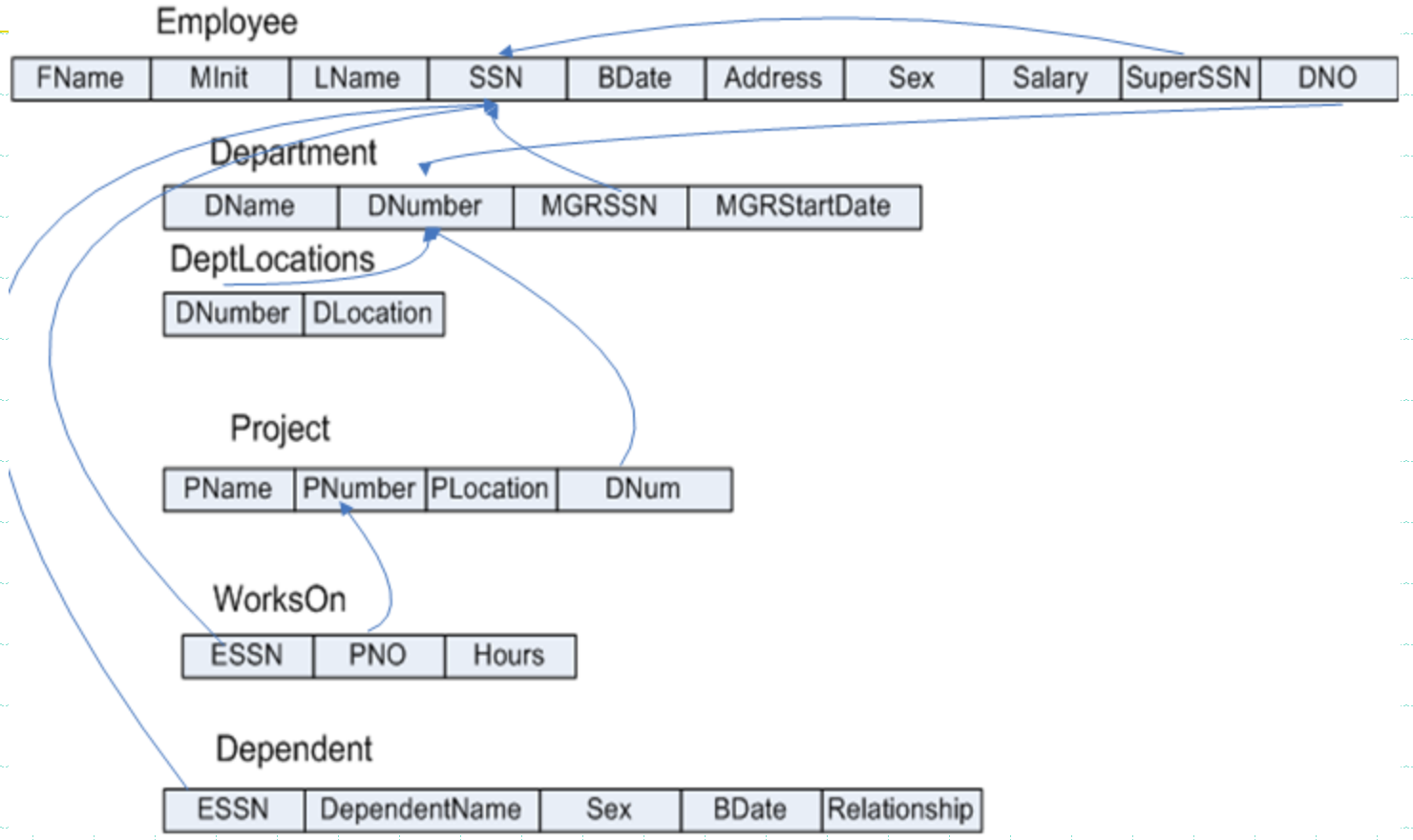
# Other Types of Constraints

Semantic Integrity Constraints:

- based on application semantics and cannot be expressed by the model
- E.g., "the max. no. of hours per employee for all projects he on is 56 hrs per week"
- A *constraint specification language* may have to be used to express these
- SQL-99 allows triggers and ASSERTIONS

# Example


Employee
FName
MInit
LName
SSN
BDate
Address
Sex
Salary
SuperSSN
DNO
Department
DName
DNumber
MGRSSN
MGRStartDate
DeptLocations
DNumber
DLocation
Project
PName
PNumber
PLocation
DNum
WorksOn
ESSN
PNO
Hours
Dependent
ESSN
DependentName
Sex
BDate
Relationship


Employee
FName
MInit
LName
SSN
BDate
Address
Sex
Salary
SuperSSN
DNO
Department
DName
DNumber
MGRSSN
MGRStartDate
DeptLocations
DNumber
DLocation
Project
PName
PNumber
PLocation
DNum
WorksOn
ESSN
PNO
Hours
Dependent
ESSN
DependentName
Sex
BDate
Relationship

# ER-to-Relational Mapping Algorithm

- **STEP 1:** For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E
- **STEP 2:** For each weak entity W with owner entity E, create a relation R, and include all simple attributes of W as attributes of R.
  - In addition, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s);

# ER-to-Relational Mapping Algorithm

- **STEP 3:** For each binary 1:1 relationship R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.
  - Choose one of the relations—S, say—and include as foreign key in S the primary key of T.
  - It is better to choose an entity type with total participation in R in the role of S. Include all the simple attributes of the 1:1 relationship type R as attributes of S

# ER-to-Relational Mapping Algorithm

- **STEP 4:** For each regular binary 1:N relationship R, identify the relation S that represents the participating entity type at the *N-side* of the relationship.
  - Include as foreign key in S the primary key of the relation T that represents the other entity participating in R.
  - Include any simple attributes of the 1:N relationship type as attributes of S.

# ER-to-Relational Mapping Algorithm

- **STEP 5:** For each binary M:N relationship R, create a new relation S to represent R.
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity; their combination will form the primary key of S.
  - include any simple attributes of the M:N relationship as attributes of S.

# ER-to-Relational Mapping Algorithm

- **STEP 6:** For each **multivalued** attribute A, create a new relation R.
  - This relation R will include an attribute corresponding to A,
  - plus the primary key attribute K—as a foreign key in R—of the relation that represents the entity or relationship that has A as an attribute.
  - The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components

# ER-to-Relational Mapping Algorithm

- **STEP 7**: For each n-ary relationship R, where n > 2, create a new relation S to represent R.
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity.
  - Also include any simple attributes of the n-ary relationship type as attributes of S.
  - The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types.

# ER vs.Relational

| ER Model | Relational Model |
| --- | --- |
| Entity type | "Entity" relation |
| 1:1 or 1:N relationship type | Foreign key (or "relationship" relation) |
| M:N relationship type | "Relationship" relation and two foreign keys |
| n-ary relationship type | "Relationship" relation and n foreign keys |
| Simple attribute | Attribute |
| Composite attribute | Set of simple component attributes |
| Multivalued attribute | Relation and foreign key |
| Value set | Domain |
| Key attribute | Primary (or secondary) key |