Unit 4
The Relational Data Model and Relational Database Constraints

- Relational Model Concepts
- Relational Model Constraints and Relational Database Schemas
- Update Operations, Transactions, and Dealing with Constraint Violations

Introduction

- The relational data model was first introduced by Ted Codd in 1970 and it attracted immediate attention due to its simplicity and mathematical foundation.
- The model uses the concept of a mathematical relation (looks like table of values) as its basic building block.
- Current popular commercial relational DBMSs (RDBMSs) include DB2 (from IBM), Oracle (from Oracle), Sybase DBMS (now from SAP), and SQLServer and Microsoft Access (from Microsoft). In addition, several open source systems, such as MySQL and PostgreSQL, are available.
- Structured Query Language (SQL) is a comprehensive model and language that is the standard for relational DBMSs.

Relational Model Concepts

- Relational model represents database as a collection of **relations**. Each relation resembles a table of values.
- When a relation is thought of as a table of values, each row in the table represents a collection of related data values. A row represents a real-world entity or relationship.
- The table name and column names are used to help to interpret the meaning of the values in each row.
- In relational model terminology, a row is called a **tuple**, a column header is called an **attribute**, and the table is called a **relation**.
- The data type describing the types of values that can appear in each column is represented by a **domain** of possible values.

Relational Model Concepts

- **Domains, Attributes, Tuples, and Relations:**
  - A **domain** D is a set of atomic values. By **atomic** we mean that each value in the domain is indivisible. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain, to help in interpreting its values. For example, *Usa_phone_numbers* represent set of ten-digit phone numbers valid in the United States.
  - A data type or format is also specified for each domain. For example, the data type for the domain Usa_phone_numbers can be declared as a character string of the form (ddd)ddd-dddd, where each d is a numeric (decimal) digit and the first three digits form a valid telephone area code.
  - A domain is thus given a name, data type, and format.

Relational Model Concepts

- Additional information for interpreting the values of a domain can also be given; for example, a numeric domain such as *Person_weights* should have the units of measurement, such as pounds or kilograms.
- A **relation schema** is denoted by R(A_1, A_2, …, A_n), is made up of a relation name R and a list of attributes, A_1, A_2, …, A_n. Each **attribute** A_i is the name of a role played by some domain D in the relation schema R. D is called the **domain** of A_i and is denoted by **dom(A_i)**. A relation schema is used to describe a relation; R is called the **name** of this relation. The **degree** (or **arity**) of a relation is the number of attributes n of its relation schema.
- A relation of degree seven would contain seven attributes describing each student as follows:

STUDENT(Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)

Relational Model Concepts

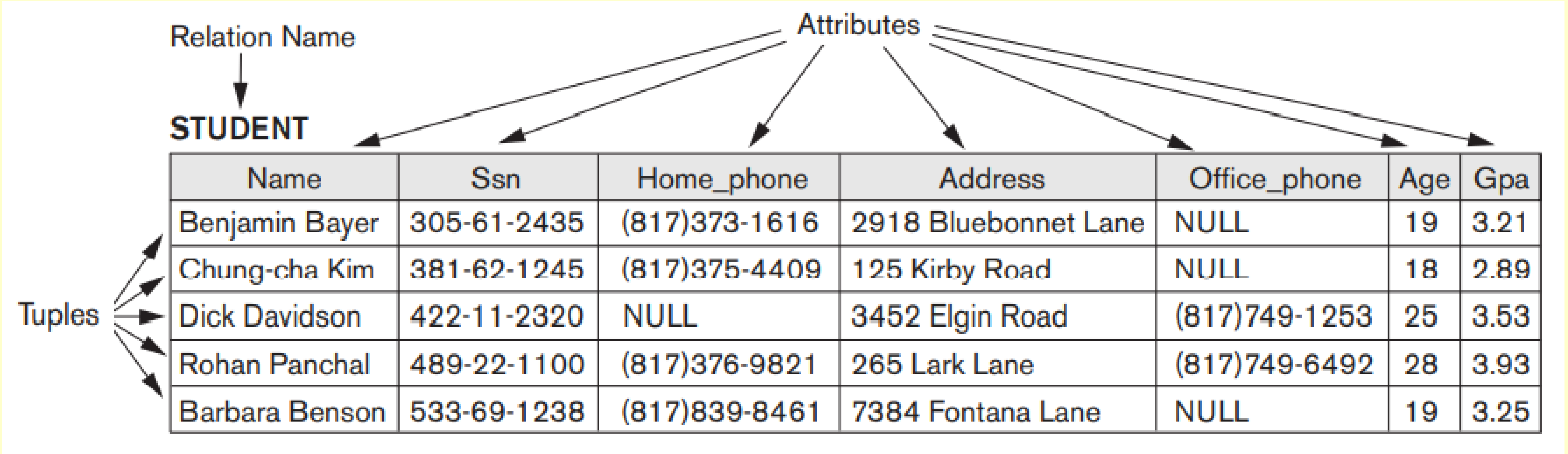
- Using the data type of each attribute, the definition is sometimes written as:

STUDENT(Name: string, Ssn: string, Home_phone: string, Address: string, Office_phone: string, Age: integer, Gpa: real)

- More precisely, we can specify previously defined domains for some of the attributes of the relation. For example, dom(HomePhone) = USA_phone_numbers. It is also possible to refer to attributes of a relation schema by their position within the relation; thus, the second attribute of the STUDENT relation is Ssn, whereas the fourth attribute is Address.
- A **relation** (or **relation state**) r of the relation schema $R(A_1, A_2, \ldots, A_n)$, also denoted by r(R), is a set of n-tuples $r = \{t_1, t_2, \ldots, t_m\}$. Each n-tuple t is an ordered list of n values $t = <v_1, v_2, \ldots, v_n>$, where each value v_i , $1 \le i \le n$, is an element of dom (A_i) or is a special NULL value.

Relational Model Concepts

- The ith value in tuple t, which corresponds to the attribute A_i , is referred to as t[A_i] or t.A_i (or t[i] if we use the positional notation).
- The terms relation intension for the schema R and relation extension for a relation state r(R) are also commonly used.
- A relation state changes frequently and relation schema is relatively static and changes very infrequently.



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

Figure The attributes and tuples of a relation STUDENT.

Relational Model Concepts

- Relation can be restated more formally using set theory. A relation (or relation state) r(R) is a mathematical relation of degree n on the domains dom(A_1), dom(A_2), …, dom(A_n), which is a subset of the Cartesian product (denoted by ×) of the domains that define R:

$$r(R) \subseteq (dom(A_1) \times dom(A_2) \times \ldots \times (dom(A_n))$$

- If we denote the total number of values, or cardinality, in a domain D by |D|, the total number of tuples in the Cartesian product is

$$|dom(A_1)| \times |dom(A_2)| \times \ldots \times |dom(A_n)|$$

- This product of cardinalities of all domains represents the total number of possible tuples that can ever exist in any relation state r(R). Of all these, a relation state at a given time reflects only the valid tuples that represent a particular state of the real world.
- It is possible for several attributes to have the same domain. The attribute names indicate different roles, or interpretations, for the domain.

Relational Model Concepts

- **Characteristics of Relations:**
  - **Ordering of Tuples in a Relation:** A relation is defined as a set of tuples. Mathematically, elements of a set have no order among them; hence, tuples in a relation do not have any particular order.
  - **Ordering of Values within a Tuple and an Alternative Definition of a Relation:** The common approach uses the ordering of values in a tuple and ordering of attributes in a relation schema. The order of attributes and their values is not that important as long as the correspondence between attributes and values is maintained.

  A more general alternative definition of relation does not require such ordering. A tuple can be considered as a set of (<attribute>, <value>) pairs, where each pair gives the value of the mapping from an attribute A_i to a value v_i from dom(A_i). It is known as self-describing data as description of each value (attribute name) is included in the tuple.

Relational Model Concepts

- **Values and NULLs in the Tuples:** Each value in a tuple is an atomic value. Hence, composite and multivalued attributes are not allowed. Hence, multivalued attributes must be represented by separate relations, and composite attributes are represented only by their simple component attributes in the basic relational model.

  NULL values are used to represent the values of attributes that may be unknown or may not apply to a tuple. In general, we can have several meanings for NULL values, such as value unknown, value exists but is not available, or attribute does not apply to this tuple.

- **Interpretation (Meaning) of a Relation:** Relation schema can be interpreted as a declaration or a type of assertion. Some relations may represent facts about entities, whereas other relations may represent facts about relationships. An alternative interpretation of a relation schema is as a predicate where the values in each tuple are interpreted as values that satisfy the predicate.

Relational Model Concepts

- **Relational Model Notation:**
  - A relation schema R of degree n is denoted by $R(A_1, A_2, \ldots, A_n)$. The uppercase letters Q, R, S denote relation names. The lowercase letters q, r, s denote relation states. The letters t, u, v denote tuples. In general, the name of a relation such as STUDENT also indicates the current relation state whereas STUDENT(Name, Ssn, …) refers only to the relation schema.
  - An attribute A can be qualified with the relation name R to which it belongs by using the dot notation R.A such as STUDENT.Name.
  - An n-tuple t in a relation r(R) is denoted by t = <v1, v2, …, vn> , where v_i is the value corresponding to attribute A_i. The following notation refers to component values of tuples: Both $t[A_i]$ and $t.A_i$ (and sometimes t[i]) refer to the value v_i in t for attribute A_i. Both $t[A_u, A_w, \ldots, A_z]$ and $t.(A_u, A_w, \ldots, A_z)$, where $A_u, A_w, \ldots, A_z$ is a list of attributes from R, refer to the subtuple of values from t corresponding to the attributes specified in the list.

Relational Model Constraints

- Constraints are the restrictions on the actual values in the database. Constraints determine which values are permissible and which are not in the database.
- Constraints are derived from the rules in the real world that the database represents. Constraints on databases can generally be divided into three main categories:
    1. **Inherent model-based or implicit constraints:** Constraints that are inherent in the data model. For example, the constraint that a relation cannot have duplicate tuples is an implicit constraint.
    2. **Schema-based or explicit constraint:** Constraints that can be directly expressed in the schemas of the data model, typically by specifying them in the DDL. For example, *domain constraints*, *key constraints*, *constraints on NULLs*, *entity integrity constraints*, and *referential integrity* are all explicit constraints.

Relational Model Constraints

3. **Application-based or semantic constraints:** These constraints must be expressed and enforced by the application programs or in some other way. These constraints are also called business rules. For example, *triggers* and *assertion* are semantic constraint.

- Another important category of constraints is **data dependencies**, which include *functional dependencies* and *multivalued dependencies*. They are used mainly for testing the "goodness" of the design of a relational database and are utilized in a process called *normalization*.
- Here, we discuss only second category, namely, schema-based constraints.

Relational Model Constraints

- **Domain Constraints:**
  - Domain constraints specify that within each tuple, the value of each attribute A must be an atomic value from the domain dom(A).
  - A data type is associated with each domain. For example, numeric data types, characters, Booleans, strings, and date are data types. Domains can also be described by a subrange of values from a data type or as an enumerated data type.
- **Key Constraints and Constraints on NULL Values:**
  - There are subsets of attributes with the property that no two tuples in any relation state have same combination of values for these attributes.
  - Suppose that we denote one such subset of attributes by SK; then for any two distinct tuples t_1 and t_2 in a relation state, we have the constraint that: $t_1[SK] \neq t_2[SK]$. Any such set of attributes SK is called a **superkey** of the relation schema R. Every relation has at least one default superkey – the set of all its attributes.

Relational Model Constraints

- A superkey with no redundancy is called a **key**. A key K is a superkey with the additional property that removing any attribute A from K leaves a set of attributes K′ that is not a superkey any more. Hence, a key satisfies two properties:
  1. Two distinct tuples in any state of the relation cannot have identical values for (all) the attributes in the key. This uniqueness property also applies to a superkey.
  2. It is a minimal superkey – that is, a superkey from which we cannot remove any attributes and still have the uniqueness constraint hold. This minimality property is required for a key but is optional for a superkey.
- Hence, a key is a superkey but not vice versa. A superkey may be a key (if it is minimal) or may not be a key (if it is not minimal).
- Any superkey formed from a single attribute is also a key. A key with multiple attributes must require all its attributes together to have the uniqueness property.

Relational Model Constraints

- The value of a key attribute can be used to identify uniquely each tuple in the relation. A key is determined from the meaning of the attributes. Key is a constraint that should hold on every valid relation state of the schema.
- A relation schema may have more than one key. In this case, each of the keys is called a **candidate key**. It is common to designate one of the candidate keys as the **primary key** of the relation. This is the candidate key whose values are used to identify tuples in the relation. We use the convention that the attributes that form the primary key of a relation schema are *underlined*.
- When a relation schema has several candidate keys, it is usually better to choose a primary key with a single attribute or a small number of attributes. The other candidate keys are designated as unique keys and are not underlined.
- Another constraint on attributes specifies whether NULL values are or are not permitted.

Relational Model Constraints

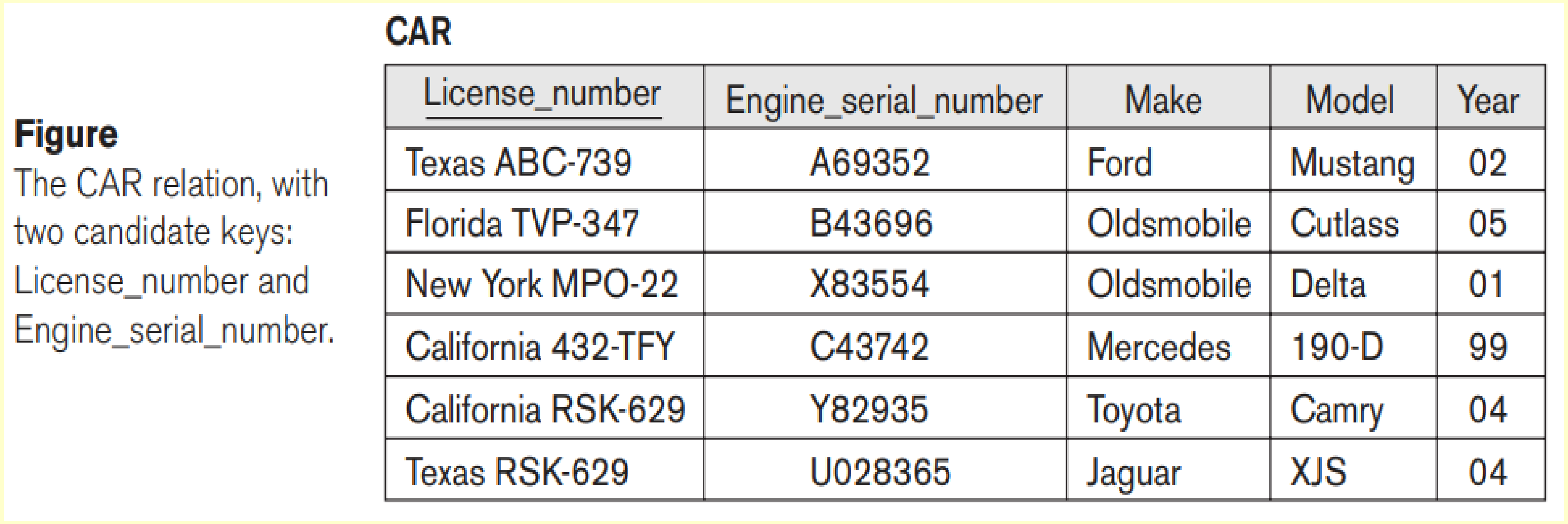


Figure
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

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

- **Entity Integrity:**
  - The entity integrity constraint states that no primary key value can be NULL. This is because the primary key value is used to identify individual tuples in a relation. Having NULL values for the primary key implies that we cannot identify some tuples.

Relational Model Constraints

- **Referential Integrity and Foreign Keys:**
  - The **referential integrity constraint** is specified between two relations and is used to maintain consistency among tuples in the two relations. The referential integrity constraint states that a tuple in one relation must refer to an existing tuple in another relation.
  - To define referential integrity more formally, first we define the concept of a **foreign key**. A set of attributes FK in relation schema R_1 is a **foreign key** of R_1 that references relation R_2 if it satisfies the following rules:
    1. The attributes in FK have the same domain(s) as the primary key attributes PK of R_2; the attributes FK are said to reference or refer to the relation R_2.
    2. A value of FK in a tuple t_1 of the current state $r_1(R_1)$ either occurs as a value of PK for some tuple t_2 in the current state $r_2(R_2)$ or is NULL. In the former case, we have $t_1[FK] = t_2[PK]$, and we say that the tuple t_1 references or refers to the tuple t_2.

Relational Model Constraints

- If the above two conditions hold, a referential integrity constraint from R_1 to R_2 is said to hold. Here, R_1 is called the referencing relation and R_2 is the referenced relation. Referential integrity also called foreign key constraint.
- Notice that a foreign key can also refer to its own relation.
- We can diagrammatically display referential integrity constraints by drawing a directed arc from each foreign key to the relation it references. For clarity, the arrowhead may point to the primary key of the referenced relation.
- All integrity constraints should be specified on the relational database schema (that is, specified as part of its definition) if we want the DBMS to enforce these constraints on the database states. Hence, the DDL includes provisions for specifying the various types of constraints so that the DBMS can automatically enforce them.

Relational Model Constraints

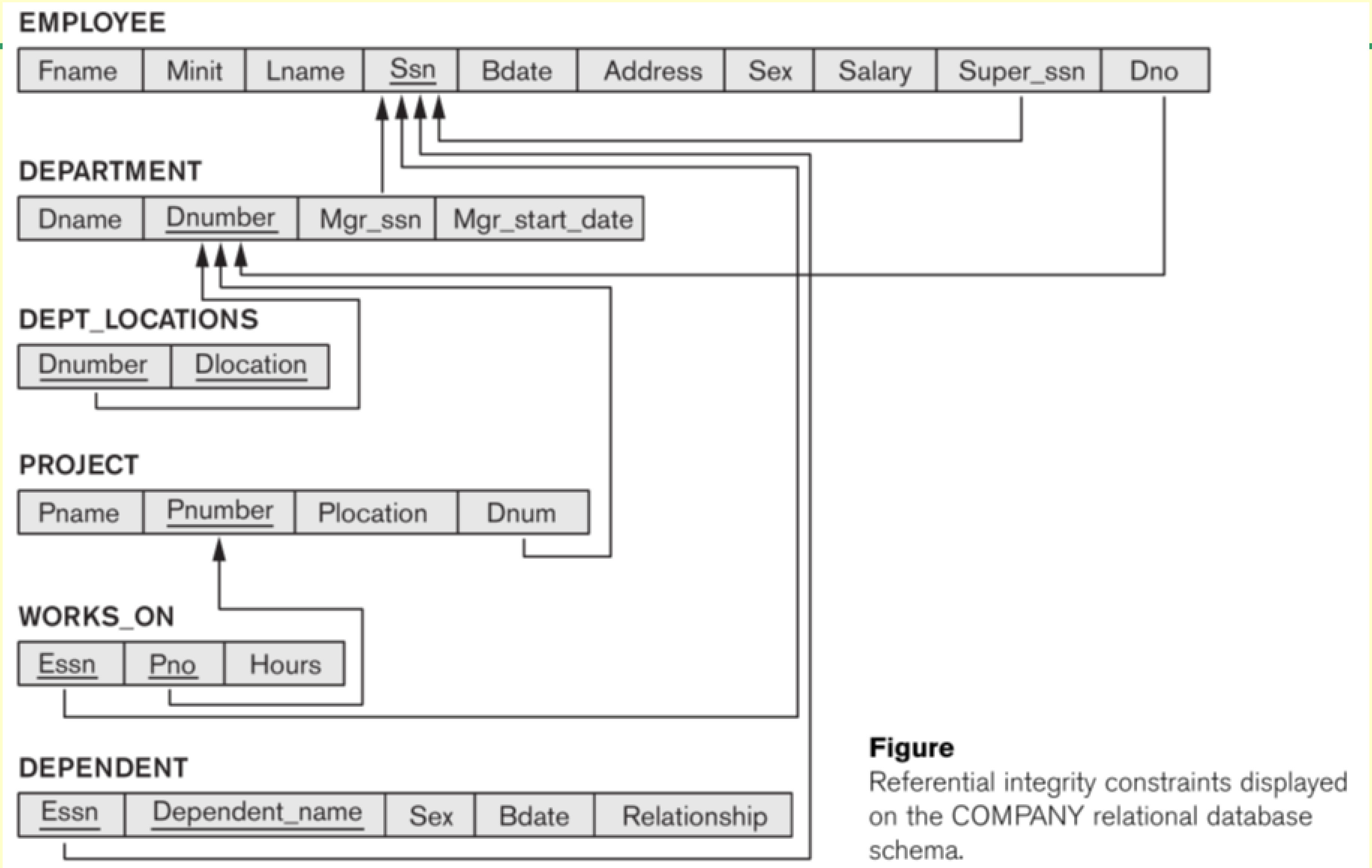


Figure
Referential integrity constraints displayed on the COMPANY relational database schema.

Relational Databases and Relational Database Schemas

- A relational database usually contains many relations, with tuples in relations that are related in various ways.
- A relational database schema S is a set of relation schemas $S = \{R_1, R_2, \ldots, R_m\}$ and a set of integrity constraints IC. A relational database state DB of S is a set of relation states $DB = \{r_1, r_2, \ldots, r_m\}$ such that each r_i is a state of R_i and such that the r_i relation states satisfy the integrity constraints specified in IC.
- A database state that does not obey all the integrity constraints is called **not valid**, and a state that satisfies all the constraints in the defined set of integrity constraints IC is called a **valid state**.

Relational Databases and Relational Database Schemas

- Each relational DBMS must have a data definition language (DDL) for defining a relational database schema. Current relational DBMSs are mostly using SQL DDL for this purpose.
- Integrity constraints are specified on a database schema and are expected to hold on every valid database state of that schema.
- When we refer to a relational database, we implicitly include both its schema and its current state.
- It is always a good practice to give distinct names for all the attributes in the database.

Update Operations and Dealing with Constraint Violations

- The operations of the relational model can be categorized into **retrievals** and **updates**. Update operations (aka modification operations) can change the states of relations in the database.
- The basic update operations are **Insert**, **Delete**, and **Update** (or **Modify**). These operations insert new data, delete old data, or modify existing data
- Insert is used to insert one or more new tuples in a relation, Delete is used to delete tuples, and Update (or Modify) is used to change values of some attributes in existing tuples.
- Whenever these operations are applied, the integrity constraints specified on the relational database schema should not be violated.

Update Operations and Dealing with Constraint Violations

- **The Insert Operation:**
  - Insert can violate any of the four types of constraints.
  - Domain constraints can be violated if an attribute value is given that does not appear in the corresponding domain or is not of the appropriate data type.
  - Key constraints can be violated if a key value in the new tuple t already exists in another tuple in the relation r(R).
  - Entity integrity can be violated if any part of the primary key of the new tuple t is NULL.
  - Referential integrity can be violated if the value of any foreign key in t refers to a tuple that does not exist in the referenced relation.
  - If an insertion violates one or more constraints, the default option is to reject the insertion, possibly with reason of rejection. Another option is to attempt to correct the reason for rejecting the insertion.

Update Operations and Dealing with Constraint Violations

- **The Delete Operation:**
  - This operation can violate only referential integrity. This occurs if the tuple being deleted is referenced by foreign keys. We specify condition on the attributes to select tuples to be deleted.
  - Several options are available to deal with constraint violation. The first option, called **restrict**, is to reject the deletion. The second option, called **cascade**, is to cascade (or propagate) the deletion by deleting tuples that reference the tuple that is being deleted. A third option, called **set null** or **set default**, is to modify the referencing attribute values that cause the violation with NULL or reference another default valid tuple. Combinations of these three options are also possible.
  - If a referencing attribute is part of the primary key, it cannot be set to NULL; otherwise, it would violate entity integrity.
  - In general, when a referential integrity constraint is specified in the DDL, the DBMS will allow the database designer to specify which of the options applies in case of a violation of the constraint.

Update Operations and Dealing with Constraint Violations

- **The Update Operation:**
  - It is necessary to specify a condition on the attributes of the relation to select the tuple (or tuples) to be modified.
  - While updating value of an attribute that is neither part of a primary key nor part of a foreign key, DBMS need only check to confirm that the new value is of the correct data type and domain.
  - Modifying a primary key value is similar to deleting one tuple and inserting another in its place. Hence, the issues discussed for Insert and Delete come into play.
  - If a foreign key attribute is modified, DBMS must ensure that the new value refers to an existing tuple in the referenced relation or set to NULL.

Transaction

- A database application program typically executes one or more transactions.
- A transaction is an executing program that includes some database operations, such as reading from the database, or applying insertions, deletions, or updates to the database. A transaction is an atomic unit of work such as bank withdrawal. At the end, transactions must leave the database in a valid or consistent state that satisfies all the constraints specified..
- A large number of commercial applications in **online transaction processing (OLTP)** systems are executing transactions at rates that reach several hundred per second.