Week 2 | Meerza Ahmed

Tuesday, August 29, 2023 1:05 PM

Lecture 2: Link | Syllabus

**Class Week 2/15*

Database Systems by Coronel & Morris

Chapter 3: The Relational Database Model

* 3.1 *A Logical View of Data*

The logical view of the relational database is facilitated by the creation of data relationships based on a logical construct known as a relation. Because a relation is a mathematical construct, end users find it much easier to think of a relation as a table

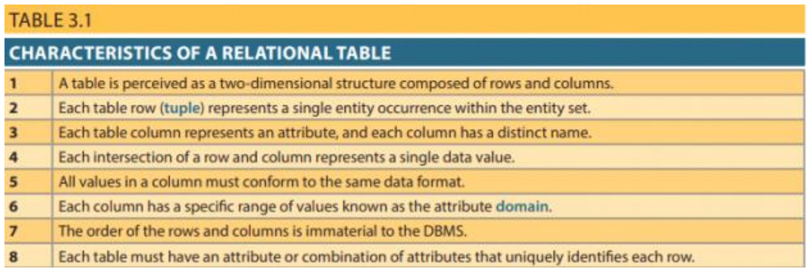


TABLE 3.1

CHARACTERISTICS OF A RELATIONAL TABLE

1	A table is perceived as a two-dimensional structure composed of rows and columns.
2	Each table row (tuple) represents a single entity occurrence within the entity set.
3	Each table column represents an attribute, and each column has a distinct name.
4	Each intersection of a row and column represents a single data value.
5	All values in a column must conform to the same data format.
6	Each column has a specific range of values known as the attribute domain.
7	The order of the rows and columns is immaterial to the DBMS.
8	Each table must have an attribute or combination of attributes that uniquely identifies each row.

* 3.2 *Keys*

In the relational model, keys are important because they are used to ensure that each row in a table is uniquely identifiable. They are also used to establish relationships among tables and to ensure the integrity of the data. A key consists of one or more attributes that determine other attributes. For example, an invoice number identifies all of the invoice attributes, such as the invoice date and the customer name

* In the relational model, a primary key is an identifier composed of one or more attributes that uniquely identifies a row. Also, a candidate key selected as a unique entity identifier

* 3.2-a *Dependencies*

The role of a key is based on the concept of determination. Determination is the state in which knowing the value of one attribute makes it possible to determine the value of another. Given any two values, you can determine the third. Determination in a database environment, however, is not normally based on a formula but on the relationships among the attributes

* A specific terminology and notation is used to describe relationships based on determination. The relationship is called functional dependence, which means that the value of one or more attributes determines the value of one or more other attributes

* The attribute whose value determines another is called the determinant or the key. The attribute whose value is determined by the other attribute is called the dependent. Using this terminology, it would be correct to say that STU_ ID is the determinant and STU_LNAME is the dependent. STU_ID functionally determines STU_LNAME, and STU_LNAME is functionally dependent on STU_ID.

* 3.2-b *Types of Keys*

A key is an attribute or group of attributes that can determine the values of other attributes. Therefore, keys are determinants in functional dependencies.

Several different types of keys are used in the relational model…

- A primary key (PK) is an attribute or combination of attributes that uniquely identifies any given row.
- A composite key is composed of more than one attribute, these attributes are known as key attributes
- A super key can uniquely identify any row in the table. In other words, a super key functionally determines every attribute in the row
- A candidate key is a minimal super key—that is, a super key without any unnecessary attributes

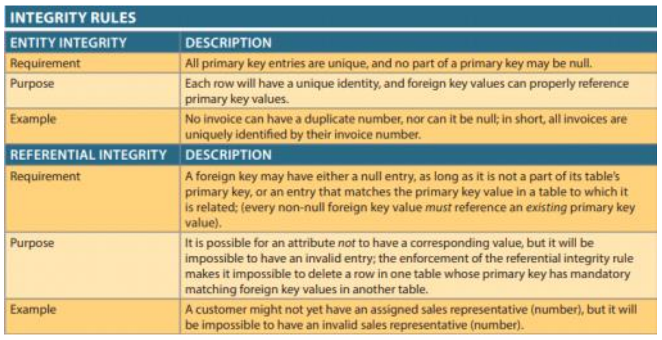
* A candidate key is based on a full functional dependency. For example, STU_NUM would be a candidate key, as would (STU_LNAME, STU_FNAME, STU_INIT, STU_PHONE). On the other hand, (STU_NUM, STU_LNAME) is a superkey, but it is not a candidate key because STU_LNAME could be removed and the key would still be a superkey.

* Entity integrity is the condition in which each row (entity instance) in the table has its own unique identity. To ensure entity integrity, the primary key has two requirements:
(1) all of the values in the primary key must be unique
(2) no key attribute in the primary key can contain a null.

* A null is the absence of an attribute value. Note that a null is not a blank
* A foreign key (FK) is the primary key of one table that has been placed into another table to create a common attribute
Foreign keys are used to ensure referential integrity, the condition in which every reference to an entity instance by another entity instance is valid. In other words, every foreign key entry must either be null or a valid value in the primary key of the related table
Finally, a secondary key is defined as a key that is used strictly for data retrieval purposes, besides the primary key
If the primary key is the customer number; the secondary key is the combination of the customer's last name and phone number. Keep in mind that a secondary key does not necessarily yield a unique outcome

* 3.2 *Integrity Rules*

Relational database integrity rules are very important to good database design. RDBMSs enforce integrity rules automatically, but it is safer to make sure your application design conforms to the entity and referential integrity rules

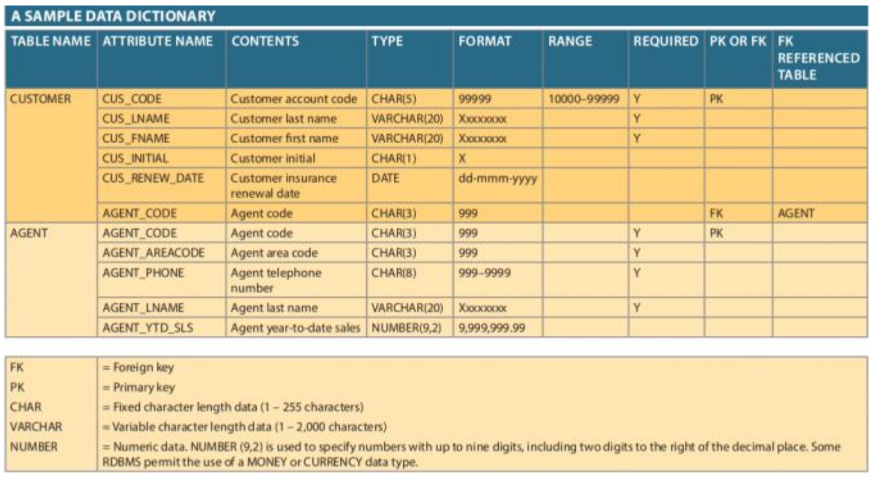


INTEGRITY RULES

ENTITY INTEGRITY	DESCRIPTION
Requirement	All primary key entries are unique, and no part of a primary key may be null.
Purpose	Each row will have a unique identity, and foreign key values can properly reference primary key values.
Example	No invoice can have a duplicate number, nor can it be null; in short, all invoices are uniquely identified by their invoice number.
REFERENTIAL INTEGRITY	**DESCRIPTION**
Requirement	A foreign key may have either a null entry, as long as it is not a part of its table's primary key, or an entry that matches the primary key value in a table to which it is related; (every non-null foreign key value *must* reference an *existing* primary key value).
Purpose	It is possible for an attribute *not* to have a corresponding value, but it will be impossible to have an invalid entry; the enforcement of the referential integrity rule makes it impossible to delete a row in one table whose primary key has mandatory matching foreign key values in another table.
Example	A customer might not yet have an assigned sales representative (number), but it will be impossible to have an invalid sales representative (number).

* To avoid nulls, some designers use special codes, known as flags, to indicate the absence of some value. Flags are special codes implemented by designers to trigger a required response, alert end users to specified conditions, or encode values. Flags may be used to prevent nulls by bringing attention to the absence of a value in a table

* 3.5 *The Data Dictionary and the System Catalog*



A SAMPLE DATA DICTIONARY

TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	RANGE	REQUIRED	PK OR FK	FK REFERENCED TABLE
CUSTOMER	CUS_CODE	Customer account code	CHAR(5)	99999	10000–99999	Y	PK	
	CUS_LNAME	Customer last name	VARCHAR(20)	Xxxxxxxx		Y		
	CUS_FNAME	Customer first name	VARCHAR(20)	Xxxxxxxx		Y		
	CUS_INITIAL	Customer initial	CHAR(1)	X				
	CUS_RENEW_DATE	Customer insurance renewal date	DATE	dd-mmm-yyyy				
	AGENT_CODE	Agent code	CHAR(3)	999			FK	AGENT
AGENT	AGENT_CODE	Agent code	CHAR(3)	999		Y	PK	
	AGENT_AREACODE	Agent area code	CHAR(3)	999		Y		
	AGENT_PHONE	Agent telephone number	CHAR(8)	999–9999		Y		
	AGENT_LNAME	Agent last name	VARCHAR(20)	Xxxxxxxx		Y		
	AGENT_YTD_SLS	Agent year-to-date sales	NUMBER(9,2)	9,999,999.99				

FK	= Foreign key
PK	= Primary key
CHAR	= Fixed character length data (1 – 255 characters)
VARCHAR	= Variable character length data (1 – 2,000 characters)
NUMBER	= Numeric data. NUMBER (9,2) is used to specify numbers with up to nine digits, including two digits to the right of the decimal place. Some RDBMS permit the use of a MONEY or CURRENCY data type.

* The data dictionary is a DBMS component that stores metadata— data about data. Thus, the data dictionary contains the data definition as well as their characteristics and relationships. A data dictionary may also include data that are external to the DBMS. Also known as an information resource dictionary. See also active data dictionary, metadata, and passive data dictionary.

* The system catalog is a detailed system data dictionary that describes all objects in a database.

* Homonyms are similar-sounding words with different meanings, such as boar and bore, or a word with different meanings, such as fair (which means "just" in some contexts and "festival" in others). In a database context, the word homonym indicates the use of the same name to label different attributes.
For example, you might use C_NAME to label a customer name attribute in a CUSTOMER table and use C_NAME to label a consultant name attribute in a CONSULTANT table. To lessen confusion, you should avoid database homonyms; the data dictionary is very useful in this regard

* A synonym is the opposite of a homonym, and indicates the use of different names to describe the same attribute. For example, car and auto refer to the same object. Synonyms must be avoided whenever possible

* 3.6 *Relationships within the relational Database*

Relationships are classified as one-to-one (1:1), one-to-many (1:M), and many-to-many (M:N or M:M)

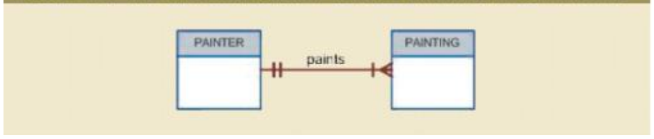
Keep in mind that…

- The 1:M relationship is the relational modeling ideal. Therefore, this relationship type should be the norm in any relational database design
- The 1:1 relationship should be rare in any relational database design
- M:N relationships cannot be implemented as such in the relational model. Any M:N relationship can be changed into two 1:M relationships

* 3.6-a *The 1:M Relationship*

The 1:M relationship is the norm for relational databases. The relationship is found in any database environment

FIGURE 3.17 THE 1:M RELATIONSHIP BETWEEN PAINTER AND PAINTING

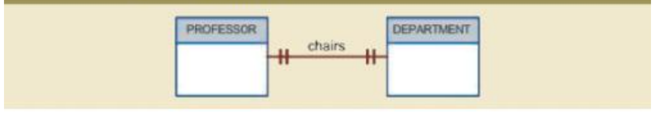


* 3.6-b *The 1:1 Relationship*

The 1:1 relationship means that one entity can be related to only one other entity, and vice versa.

For example, one department chair—a professor—can chair only one department, and one department can have only one department chair

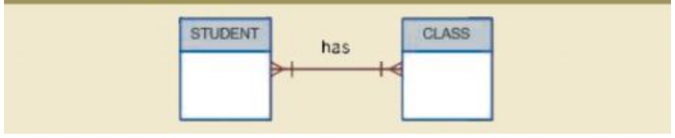
FIGURE 3.21 THE 1:1 RELATIONSHIP BETWEEN PROFESSOR AND DEPARTMENT



* 3.6-c *The M:N Relationship*

A many-to-many (M:N) relationship is not supported directly in the relational environment. However, M:N relationships can be implemented by creating a new entity in 1:M relationships with the original entities

FIGURE 3.23 THE ERM'S M:N RELATIONSHIP BETWEEN STUDENT AND CLASS

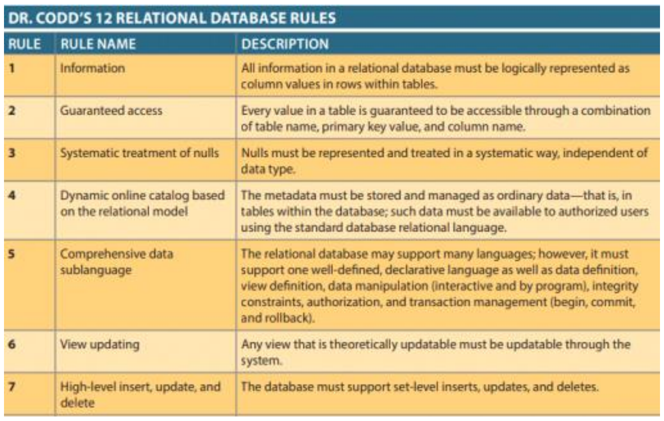
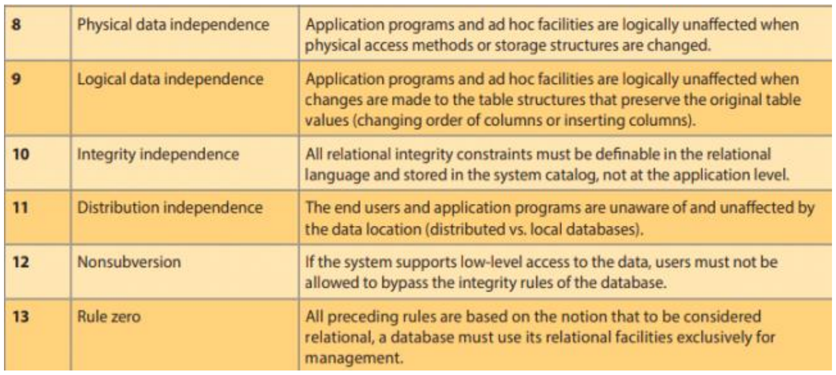


* A problems inherent in the many-to-many relationship can easily be avoided by creating a composite entity. Because such a table is used to link the tables that were originally related in an M:N relationship, the composite entity structure includes—as foreign keys—at least the primary keys of the tables that are to be linked. The database designer has two main options when defining a composite table's primary key:
use the combination of those foreign keys or create a new primary key.

* 3.8 *Indexes*

- An Index is an ordered array of index key values and row ID values (pointers). Indexes are generally used to speed up and facilitate data retrieval.
- A index key is the index's reference point. More formally, an index is an ordered arrangement of keys and pointers. Each key points to the location of the data identified by the key
- A unique index, as its name implies, is an index key that can only have one pointer value (row) associated with it

* 3.9 *Codd's Relational Database Rules*

DR. CODD'S 12 RELATIONAL DATABASE RULES

RULE	RULE NAME	DESCRIPTION
1	Information	All information in a relational database must be logically represented as column values in rows within tables.
2	Guaranteed access	Every value in a table is guaranteed to be accessible through a combination of table name, primary key value, and column name.
3	Systematic treatment of nulls	Nulls must be represented and treated in a systematic way, independent of data type.
4	Dynamic online catalog based on the relational model	The metadata must be stored and managed as ordinary data—that is, in tables within the database; such data must be available to authorized users using the standard database relational language.
5	Comprehensive data sublanguage	The relational database may support many languages; however, it must support one well-defined, declarative language as well as data definition, view definition, data manipulation (interactive and by program), integrity constraints, authorization, and transaction management (begin, commit, and rollback).
6	View updating	Any view that is theoretically updatable must be updatable through the system.
7	High-level insert, update, and delete	The database must support set-level inserts, updates, and deletes.
8	Physical data independence	Application programs and ad hoc facilities are logically unaffected when physical access methods or storage structures are changed.
9	Logical data independence	Application programs and ad hoc facilities are logically unaffected when changes are made to the table structures that preserve the original table values (changing order of columns or inserting columns).
10	Integrity independence	All relational integrity constraints must be definable in the relational language and stored in the system catalog, not at the application level.
11	Distribution independence	The end users and application programs are unaware of and unaffected by the data location (distributed vs. local databases).
12	Nonsubversion	If the system supports low-level access to the data, users must not be allowed to bypass the integrity rules of the database.
13	Rule zero	All preceding rules are based on the notion that to be considered relational, a database must use its relational facilities exclusively for management.

Chapter 4: Entity Relationship (ER) Modeling

* 4.1 *The Entity Relationship Model (ERM)*

A conceptual models such as the ERM can be used to understand and design the data requirements of an organization. Therefore, the ERM is independent of the database type. Conceptual models are used in the conceptual design of databases, while relational models (ERD) are used in the logical design of databases

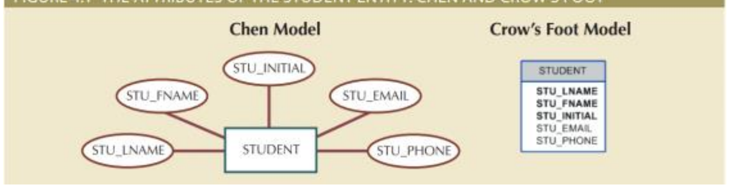
* 4.1-a *Entities*

At the ER modeling level, an entity actually refers to the entity set and not to a single entity occurrence. In other words, an entity in the ERM corresponds to a table—not to a row—in the relational environment. The ERM refers to a table row as an entity instance or entity occurrence.

* 4.1-b *Attributes*

Attributes are characteristics of entities. For example, the STUDENT entity includes the attributes STU_LNAME, STU_FNAME, and STU_INITIAL, among many others.

FIGURE 4.1 THE ATTRIBUTES OF THE STUDENT ENTITY: CHEN AND CROW'S FOOT



* A required attribute must have a value; in other words, it cannot be left empty. As shown in Figure 4.1, the two boldfaced attributes in the Crow's Foot notation indicate that data entry will be required. STU_LNAME and STU_FNAME require data entries because all students are assumed to have a last name and a first name

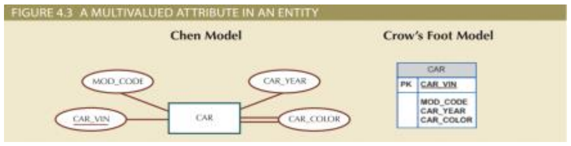
* An optional attribute is an attribute that does not require a value; therefore, it can be left empty
* Domains attributes have a domain, a domain is the set of possible values for a given attribute. The domain for a gender attribute consists of only two possibilities: M or F (or some other equivalent code).

* Identifiers (Primary Keys) The ERM uses identifiers—one or more attributes that uniquely identify each entity instance. In the relational model, entities are mapped to tables, and the entity identifier is mapped as the table's primary key (PK). Identifiers are underlined in the ERD, this also applies to all identifiers in a composite identifier
* A composite identifier, a primary key composed of more than one attribute

* A composite attribute is not the same as a composite identifier. It's an attribute that can be further subdivided to yield additional attributes. For example, a phone number such as 615-898- 2368 may be divided into an area code (615), an exchange number (898), and a four-digit code (2368)
* A simple attribute is an attribute that cannot be subdivided. For example, age, sex, and marital status would be classified as simple attributes. To facilitate detailed queries, it is wise to change composite attributes into a series of simple attributes
* A single-valued attribute is an attribute that can have only a single value. For example, a person can have only one Social Security number, and a manufactured part can have only one serial number. Keep in mind that a single-valued attribute is not necessarily a simple attribute. For instance, a part's serial number (such as SE-08-02-189935) is single-valued, but it is a composite attribute because it can be subdivided

* A multivalued attributes are attributes that can have many values. For instance, a person may have several college degrees, and a household may have several different phones, each with its own number. Similarly, a car's color may be subdivided into many colors for the roof, body, and trim. In the Chen ERM, multivalued attributes are shown by a double line connecting the attribute to the entity. The Crow's Foot notation does not identify multivalued attributes

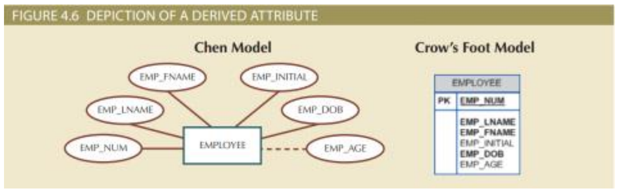
FIGURE 4.3 A MULTIVALUED ATTRIBUTE IN AN ENTITY

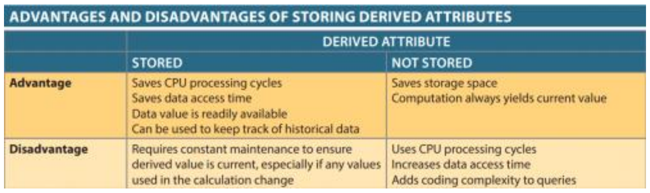


* Although the conceptual model can handle M:N relationships and multivalued attributes, you should not implement them in the RDBMS. Remember from Chapter 3 that in the relational table, each column and row intersection represents a single data value

* A derived attribute is an attribute whose value is calculated (derived) from other attributes. The derived attribute need not be physically stored within the database; instead, it can be derived by using an algorithm. For example, an employee's age, EMP_AGE, may be found by computing the integer value of the difference between the current date and the EMP_DOB

FIGURE 4.6 DEPICTION OF A DERIVED ATTRIBUTE





ADVANTAGES AND DISADVANTAGES OF STORING DERIVED ATTRIBUTES

	DERIVED ATTRIBUTE	
	STORED	NOT STORED
Advantage	Saves CPU processing cycles Saves data access time Data value is readily available Can be used to keep track of historical data	Saves storage space Computation always yields current value
Disadvantage	Requires constant maintenance to ensure derived value is current, especially if any values used in the calculation change	Uses CPU processing cycles Increases data access time Adds coding complexity to queries

* **4.1-c** *Relationships*

The entities that participate in a relationship are also known as participants, and each relationship is identified by a name that describes the relationship. The relationship name is an active or passive verb; for example, a STUDENT takes a CLASS, a PROFESSOR teaches a CLASS.