Relational Database Model

Logical View of Data

Tables and Their Characteristics

- In relational database, a relation represents the logical view of the data.
- E. F. Codd (creator of the relational model) use the term relation as a synonym for table.
- A table is perceived as a two-dimensional structure composed of rows and columns.
- A table is a persistent representation of a logical relation.
- A table contains a group of related entity occurrences – an entity set.

Table 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 row/column intersection 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 a combination of attributes that uniquely identifies each row.

Keys

. . .

- Keys are important - they are used to ensure that each row in a table is uniquely identifiable.
- 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.
- The role of a key is based on the concept known as determination.
- In the context of a database table, the statement ***"A determines B"*** indicates that if you know the value of attribute A, you can look up (determine) the value of attribute B. – *A is the key and B represents the other data related to A*.

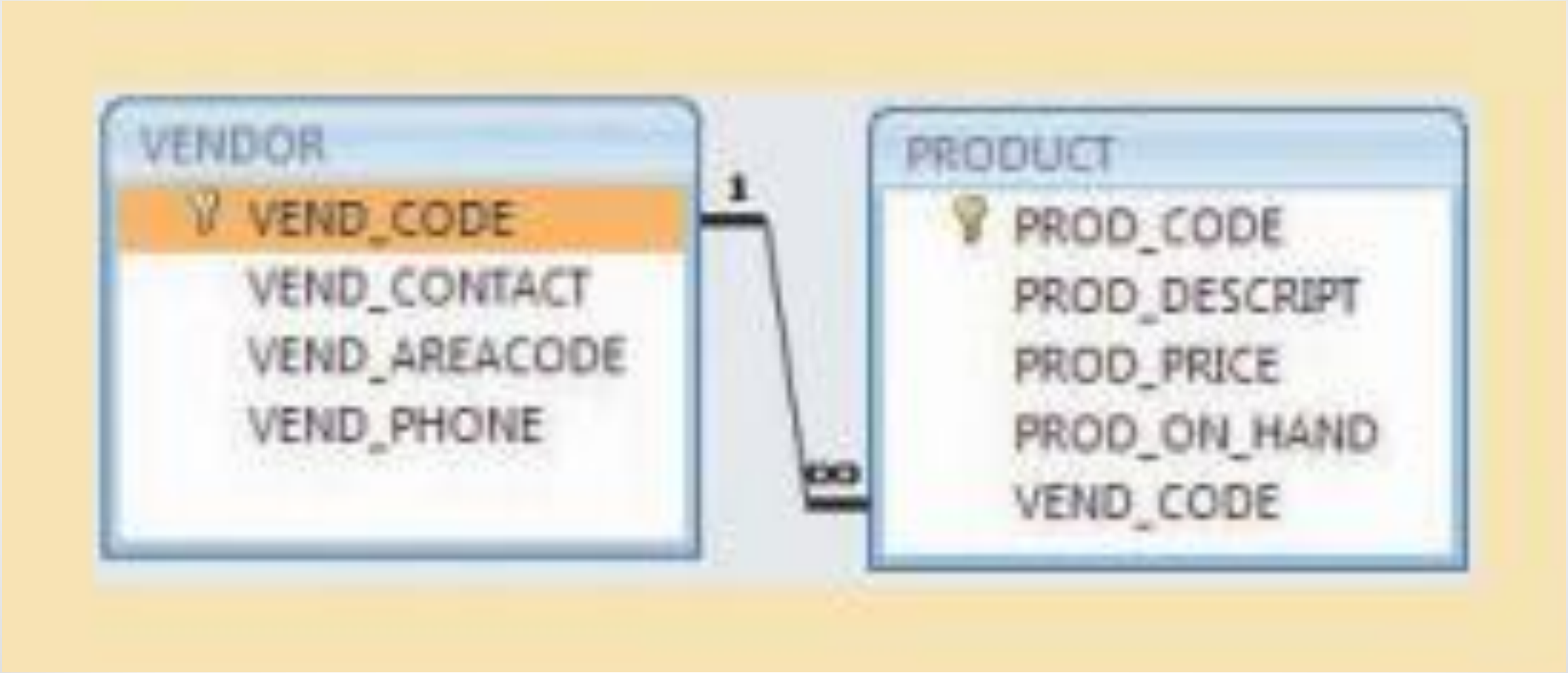
- The shorthand notation for "A determines B" is A → B
  - "**EMP_ID determines EMP_LASTNAME**"; ***EMP_ID → EMP_LASTNAME***
  - "**EMP_ID determines EMP_LASTNAME, EMP_FIRSTNAME, EMP_DEGREE**" is written as ***EMP_ID → EMP_LASTNAME, EMP_FIRSTNAME, EMP_DEGREE***
  - However, EMP_DEGREE → EMP_ID does not happen.
- Functional dependence: *Attribute B is functionally dependent on attribute A if each value in column A determines* ***one and only one*** *value in column B.*

- Composite key - two or more attributes used to define functional dependence.
- Key attribute is any attribute that is part of a key.
  - EMP_LASTNAME, EMP_FIRSTNAME, EMP_MIDDLE → EMP_DEGREE, EMP_POSITION
- Full functional dependence: *If the attribute (B) is functionally dependent on a composite key (A) but not on any subset of that composite key, the attribute (B) is fully functionally dependent on (A).*
- Superkey is any key that uniquely identifies each row
  - It functionally determines all of a row's attributes.

- Candidate key is a superkey without unnecessary attributes.
  - *EMP_ID* — *candidate key*
  - *EMP_ID, EMP_LNAME* — *not candidate key*
  - *EMP_LNAME, EMP_FNAME, EMP_MI* — *candidate key*
- The primary key *is the candidate key chosen to be the unique row identifier*.
  - Note, incidentally, that a primary key *is a superkey as well as a candidate key*.
- The choice of a primary key would be driven by the designers choice or by the end-user requirements.

- Each value of the primary key must be unique to ensure that each row is uniquely identified by the primary key – entity integrity.
- Inorder to maintain entity integrity, a primary key must not be allowed to contain a ***null*** (no data entry) value.
  - Null could mean:
    - *An unknown attribute value,*
    - *A known, but missing attribute value*
    - *A "not applicable" condition*

- A relational database can also be represented by a relational schema.
- Relational schema is a textual representation of the database tables where:
  - each table is listed by its name followed by its attributes enclosed in parenthesis.
  - The primary key(s) is/are underlined
  - ITEM(<u>ITEM_NUM</u>, ITEM_NAME,ITEM_PRICE, ITEM_QTY)
  - STUDENT(<u>ST_ID</u>, ST_LNAME,ST_FNAME,ST_MI,ST_DEGREE)
- Foreign key (FK) is an attribute whose value match the primary key values in the related value.


VENDOR
VEND_CODE
VEND_CONTACT
VEND_AREACODE
VEND_PHONE
1
PRODUCT
PROD_CODE
PROD_DESCRIPT
PROD_PRICE
PROD_ON_HAND
VEND_CODE

- Referential integrity *exist when the foreign key contains a value that refers to an existing valid tuple in another relation*.
- Secondary key *is defined as a key the is used strictly for data retrieval purposes.*

Integrity Rules

. . .

- All modern DBMSs automatically implement integrity rules.
- However it is a good design practice to have 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).

Data Dictionary and the System Catalog

- Data dictionary provides a detailed description of all tables found within the user/designer-created database.
  - Sometimes described as the database designer's database
- The system catalog can be described as a detailed system data dictionary that describes all objects within the database, including:
  - data about table names,
  - the table's creator and creation date,
  - the number of columns in each table,
  - the data type corresponding to each column,
  - index filenames,
  - index creators,
  - authorized users, and
  - access privileges.

- The systems catalog automatically produces the database documentation.
- This documentation is used to check and eliminate homonyms and synonyms as new tables are created.
- In database context:
  - *Homonyms* indicates the use of the same attribute name to label different attributes
  - *Synonym* indicates the user of different names to describe the same attributes
  - Both must be avoided.

Relationships Within the Relational Database

- One to Many
- One to One
- Many to Many
  - To avoid many to many relationships create a composite entity (bridge or associative)
  - The composite entity's foreign key at the very least should include at the primary keys of the tables to be link. It may include other attributes.

Codd's 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 On-Line 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 with support for 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 column 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 and unaffected by the data location (distributed vs. local databases).
12	**Nonsubversion**	If the system supports low-level access to the data, there must not be a way to bypass the integrity rules of the database.
	Rule Zero	All preceding rules are based on the notion that in order for a database to be considered relational, it must use its relational facilities exclusively to manage the database.

End of Presentation

Source:

Rob, Peter and Coronel, Carlos. *Database Systems : Design, Implementation and Management*, 7th Edition. Course Technology, Thomson Learning Inc.©2007