Department of Commerce (CA)

CORE PAPER-XII-DIRECT TAX

SEMESTER:I **SUB CODE: 18MCC12C**

M.COM(CA)

UNIT2: Relational approach-relational data structure-relation-domain-attributes-keys-relational algebra-introduction-traditional set operations-special set operations.

REFERENCE BOOK:

An introduction to database system-C.J. Dates

An introduction to database system-Bipin

PREPARED BY: DR. E.N. KANJANA,

ASST PROFESSOR.

Relational Model

The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name.

The data is arranged in a relation which is visually represented in a two dimensional table. The data is inserted into the table in the form of tuples (which are nothing but rows). A tuple is formed by one or more than one attributes, which are used as basic building blocks in the formation of various expressions that are used to derive a meaningful information. There can be any number of tuples in the table, but all the tuple contain fixed and same attributes with varying values. The relational model is implemented in database where a relation is represented by a table, a tuple is represented by a row, an attribute is represented by a column of the table, attribute name is the name of the column such as 'identifier', 'name', 'city' etc., attribute value contains the value for column in the row. Constraints are applied to the table and form the logical schema. In order to facilitate the selection of a particular row/tuple from the table, the attributes i.e. column names are used, and to expedite the selection of the rows some fields are defined uniquely to use them as indexes, this helps in searching the required data as fast as possible. All the relational algebra operations, such as Select, Intersection, Product, Union, Difference, Project, Join, Division, Merge etc. can also be performed on the Relational Database Model. Operations on the Relational Database Model are facilitated with the help of different conditional expressions, various key attributes, pre-defined constraints etc.

Keys:

Keys are attributes or set of attributes used to distinguish one entity from another in an entity set.

1) **Super Key:** A super key is set of one or more attributes that can uniquely identify an entity in an entity set.

2) **Candidate Key:** All the attributes or set of attribute, when can uniquely identify an entity are candidate keys. Only those key can be candidate key whose no proper subset is a superkey.

3) **Primary Key:** The primary key is the term used for the candidates key that is chosen by the database designer as the principal means of identifying an entity.

4) **Alternate Keys:** The alternate key is term used for the candidate keys that are remaining after the primary key has be choosen by database designer.
5) **Foreign Key:** A foreign key is an attribute or set of attribute in a relation of database that serve as the primary key of another relation in the same database.
6) **Composite Key:** A primary key that consists of more than one attribute is called composite key

Database Languages

A database system provides a **data definition language** to specify the database schema and a **data manipulation language** to express database queries and updates. In practice, the data definition and data manipulation languages are not two separate languages; instead they simply form parts of a single database language, such as the widely used SQL language.

1.9.1 Data-Definition Language

We specify a database schema by a set of definitions expressed by a special language called a **data-definition language** (**DDL**).

For instance, the following statement in the SQL language defines the *account* table:

create table *account (account-number* ***char****(10), balance* ***integer****)*

Execution of the above DDL statement creates the *account* table. In addition, it updates a special set of tables called the **data dictionary** or **data directory**.

A data dictionary contains **metadata**—that is, data about data. The schema of a table is an example of metadata. A database system consults the data dictionary before reading or modifying actual data.

We specify the storage structure and access methods used by the database system by a set of statements in a special type of DDL called a **data storage and definition** language. These statements define the implementation details of the database schemas, which are usually hidden from the users.

The data values stored in the database must satisfy certain **consistency constraints**. For example, suppose the balance on an account should not fall below $100. The DDL provides facilities to specify such constraints. The database systems check these constraints every time the database is updated.

1.9.2 Data-Manipulation Language

Data manipulation is

The retrieval of information stored in the database

The insertion of new information into the database

The deletion of information from the database

The modification of information stored in the database

A **data-manipulation language (DML)** is a language that enables users to access or manipulate data as organized by the appropriate data model. There are basically two types:

Procedural DMLs require a user to specify *what* data are needed and *how* to get those data.

Declarative DMLs (also referred to as **nonprocedural** DMLs) require a user to specify *what* data are needed *without* specifying how to get those data.
Declarative DMLs are usually easier to learn and use than are procedural DMLs. However, since a user does not have to specify how to get the data, the database system has to figure out an efficient means of accessing data. The DML component of the SQL language is nonprocedural.

Relational Model Concepts
We shall represent a relation as a table with columns and rows. Each column of the **table** has a name, or ***attribute***. Each row is called a ***tuple***.
• **Domain**: a set of atomic values that an attribute can take
• **Attribute**: name of a column in a particular table (all data is stored in tables).
Each attribute Ai must have a *domain*, dom(A_i).
• **Relational Schema**: The design of one table, containing the name of the table (i.e. the name of the relation), and the names of all the columns, or attributes.
Example: STUDENT(Name, SID, Age, GPA)
• **Degree of a Relation**: the number of attributes in the relation's schema.
• **Tuple, t, of R(A1, A2, A3, …, An):** an ORDERED set of values, < v1, v2, v3, …, vn>, where each v_i is a value from dom(A_i).
2
• **Relation Instance, r(R):** a set of tuples; thus, r(R) = { t1, t2, t3, …, tm}

Properties of relations
Properties of database relations are:
• relation name is distinct from all other relations
• each cell of relation contains exactly one atomic (single) value
• each attribute has a distinct name
• values of an attribute are all from the same domain
• order of attributes has no significance
• each tuple is distinct; there are no duplicate tuples
• order of tuples has no significance, theoretically.

3.2.2 Relational keys
There are two kinds of keys in relations. The first are identifying keys: the **primary key** is the main concept, while two other keys – **super key** and **candidate key** – are related concepts. The second kind is the foreign key.
3
Identity Keys
Super Keys
A super key is a set of attributes whose values can be used to uniquely identify a tuple within a relation. A relation may have more than one super key, but it always has at least one: the set of all attributes that make up the relation.
Candidate Keys
A candidate key is a super key that is minimal; that is, there is no proper subset that is itself a superkey. A relation may have more than one candidate key, and the different

candidate keys may have a different number of attributes. In other words, you should not interpret 'minimal' to mean the super key with the fewest attributes.
A candidate key has two properties:
(i) in each tuple of R, the values of K uniquely identify that tuple (uniqueness)
(ii) no proper subset of K has the uniqueness property (irreducibility).

Primary Key

The primary key of a relation is a candidate key especially selected to be the key for the relation. In other words, it is a choice, and there can be only one candidate key designated to be the primary key.

Relationship between identity keys

The relationship between keys:
Superkey ⊇ Candidate Key ⊇ Primary Key

Foreign keys

The attribute(s) within one relation that matches a candidate key of another relation. A relation may have several foreign keys, associated with different target relations.
Foreign keys allow users to link information in one relation to information in another relation. Without FKs, a database would be a collection of unrelated tables.

3.3 Relational Model Constraints

Integrity Constraints

Each relational schema must satisfy the following four types of constraints.

A. Domain constraints

Each attribute **Ai** must be an atomic value from dom(**Ai**) for that attribute.
4
The attribute, Name in the example is a BAD DESIGN (because sometimes we may want to search a person by only using their last name.

B. Key Constraints

Superkey of R: A set of attributes, SK, of R such that no two tuples in any valid relational instance, r(R), will have the same value for SK. Therefore, for any two distinct tuples, t1 and t2 in r(R),
t1[SK] != t2[SK].
Key of R: A minimal superkey. That is, a superkey, K, of R such that the removal of ANY attribute from K will result in a set of attributes that are not a superkey.
Example CAR(State, LicensePlateNo, VehicleID, Model, Year, Manufacturer)
This schema has two keys:
K1 = { State, LicensePlateNo}
K2 = { VehicleID }
Both K1 and K2 are superkeys.
K3 = { VehicleID, Manufacturer} is a superkey, but not a key (Why?).
If a relation has more than one keys, we can select any one (arbitrarily) to be the primary key. Primary Key attributes are underlined in the schema:
CAR(State, LicensePlateNo, VehicleID, Model, Year, Manufacturer)

C. Entity Integrity Constraints

The primary key attribute, PK, of any relational schema R in a database cannot have null

values in any tuple. In other words, for each table in a DB, there must be a key; for each key, every row in the table must have non-null values. This is because PK is used to identify the individual tuples.
Mathematically, t[PK] != NULL for any tuple t € r(R).

D. Referential Integrity Constraints

Referential integrity constraints are used to specify the relationships between two relations in a database.
Consider a referencing relation, R1, and a referenced relation, R2. Tuples in the referencing relation, R1, have attributed FK (called **foreign key** attributes) that reference
5
the primary key attributes of the referenced relation, R2. A tuple, t1, in R1 is said to reference a tuple, t2, in R2 if t1[FK] = t2[PK].
A referential integrity constraint can be displayed in a relational database schema as a directed arc from the referencing (foreign) key to the referenced (primary) key.

Relational Algebra

Relational algebra is a *procedural language* consisting of a set of *operators*. Each operator takes one or more relations as its input and produces *one* relation as its output. The seven basic relational algebra operations are **Selection**, **Projection**, **Joining**, **Union**, **Intersection**, **Difference** and **Division**. It is important to note that these operations do not alter the database. The relation produced by an operation is available to the user but it is not stored in the database by the operation.

Selection (also called *Restriction*)

The SELECT operator selects all tuples from some relation, so that some attributes in each tuple satisfy some condition. A new relation containing the selected tuples is then created as output. Suppose we have the relation STORES:

Projection

The projection operator constructs a new relation from some existing relation by selecting only specified attributes of the existing relation and eliminating duplicate tuples in the newly formed relation. For example,

R3 = PROJECT STORES OVER Store-ID, Location

Joining

Joining is a operation for combining two relations into a single relation. At the outset, it requires choosing the attributes to match the tuples in each relation. Tuples in different relations but with the same value of matching attributes are combined into a single tuple in the output relation.

Joining relations together based on equality of values of common attributes is called an *equijoin*. Conditions of join may be other than equality - we may also have a ‘greaterthan’ or ‘less-than’ join.
When duplicate attributes are removed from the result of an equijoin this is called an **natural join**. The example above is such a natural join - as Store-ID appears only once in

the result.
Note that there is often a connection between keys (primary and foreign) and the attributes over which a join is performed in order to amalgamate information from multiple related tables in a database. In the above example, ITEMS.Store_ID is a foreign key reflecting the primary key STORE.Store_ID. When we join on Store_ID the relationship between the tables is expressed explicitly in the resulting output table.

The **UNION** operator builds a relation consisting of all tuples appearing in either or both of two specified relations.
The **INTERSECT** operator builds a relation consisting of all tuples appearing strictly in both specified relations.
The **DIFFERENCE** operator builds a relation consisting of all tuples appearing in the first, but not the second of two specified relations.

Division
In its simplest form, this operation has a binary relation R(X,Y) as the dividend and a divisor that includes Y. The output is a set, *S,* , of values of X such that $x \in S$ if there is a row (*x,y*) in R for each *y* value in the divisor

Relational Set Operators

- This section describes the basic data manipulation

Relational Algebra

- Defines the theoretical way of manipulating table contents using the **eight relational operators.**

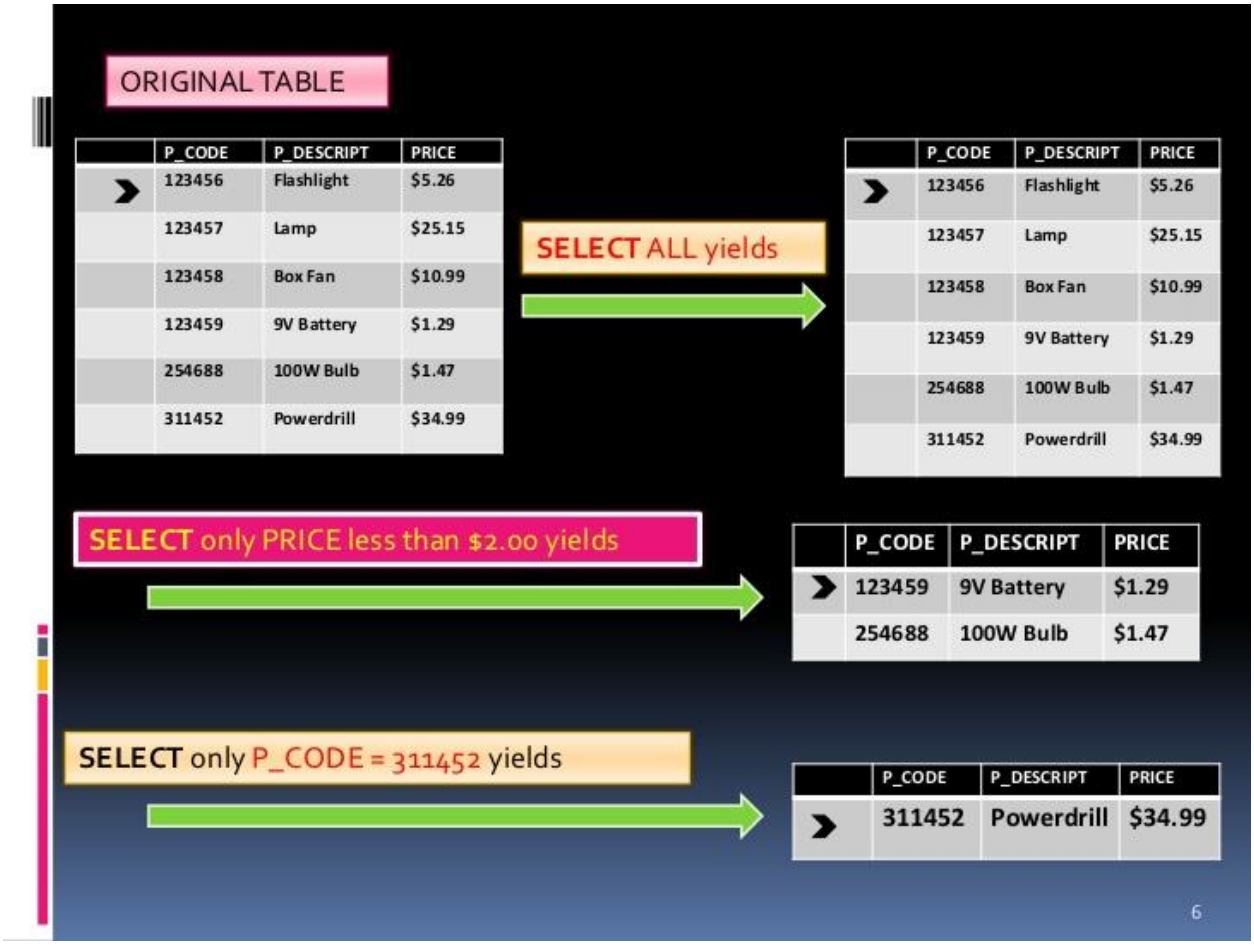
Eight Relational Operators

- SELECT
- PROJECT
- JOIN
- INTERSECT
- UNION
- DIFFERENCE
- PRODUCT
- DIVIDE

- The relational operators have the property of closure; that is, the use of relational algebra operators on existing relations (tables) produces new relations.
- There is no need to examine the mathematical definitions, properties and characteristics of those relational algebra operators.

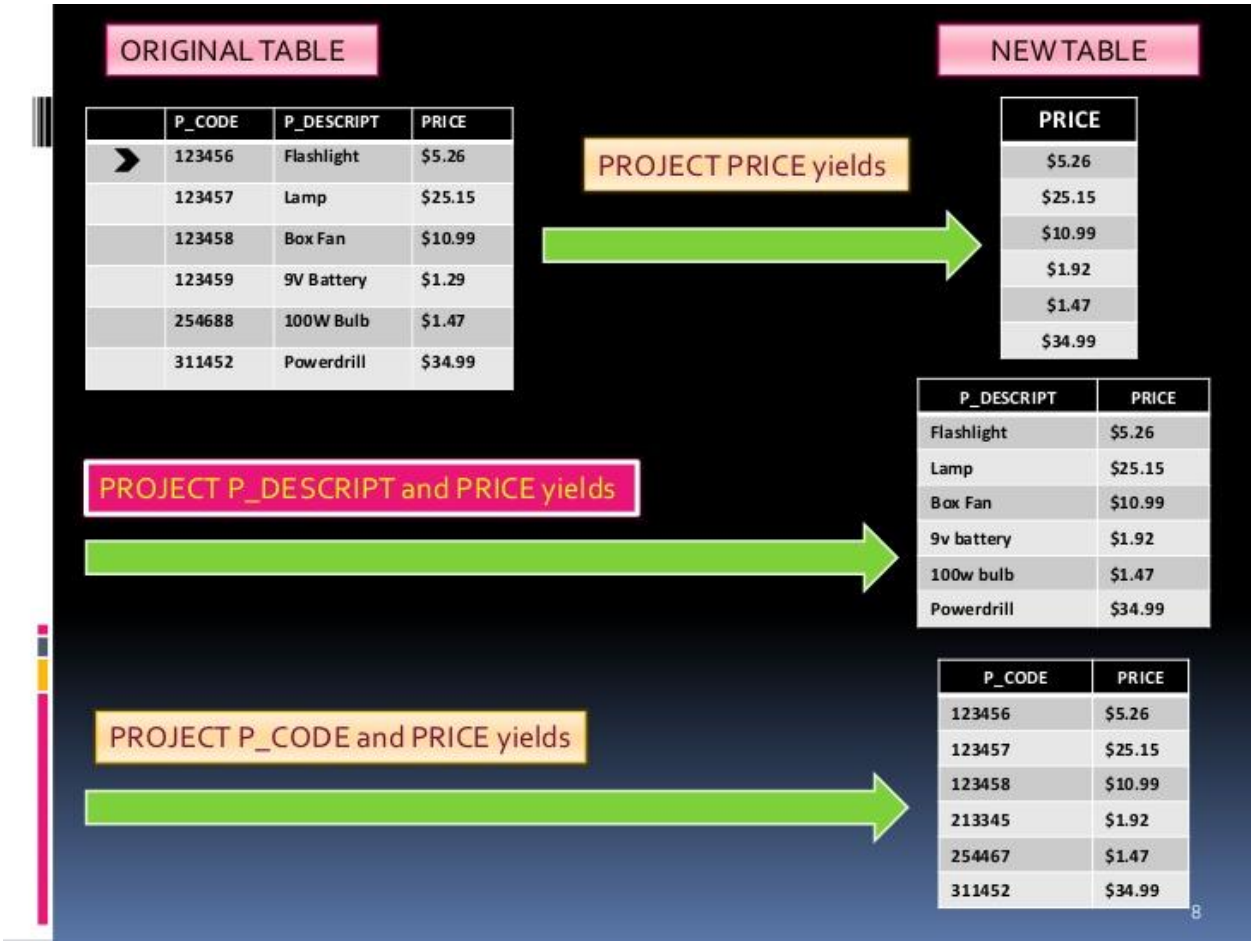
SELECT

- Also known as RESTRICT
- Yields values for all rows found in a table.
- Used to list all of the row values, or can yield only those row values that match a specified criterion.
- SELECT yields a horizontal subset of a table.


ORIGINAL TABLE
P_CODE P_DESCRIPT PRICE
123456 Flashlight $5.26
123457 Lamp $25.15
123458 Box Fan $10.99
123459 9V Battery $1.29
254688 100W Bulb $1.47
311452 Powerdrill $34.99
SELECT ALL yields
P_CODE P_DESCRIPT PRICE
123456 Flashlight $5.26
123457 Lamp $25.15
123458 Box Fan $10.99
123459 9V Battery $1.29
254688 100W Bulb $1.47
311452 Powerdrill $34.99
SELECT only PRICE less than $2.00 yields
P_CODE P_DESCRIPT PRICE
123459 9V Battery $1.29
254688 100W Bulb $1.47
SELECT only P_CODE = 311452 yields
P_CODE P_DESCRIPT PRICE
311452 Powerdrill $34.99
6

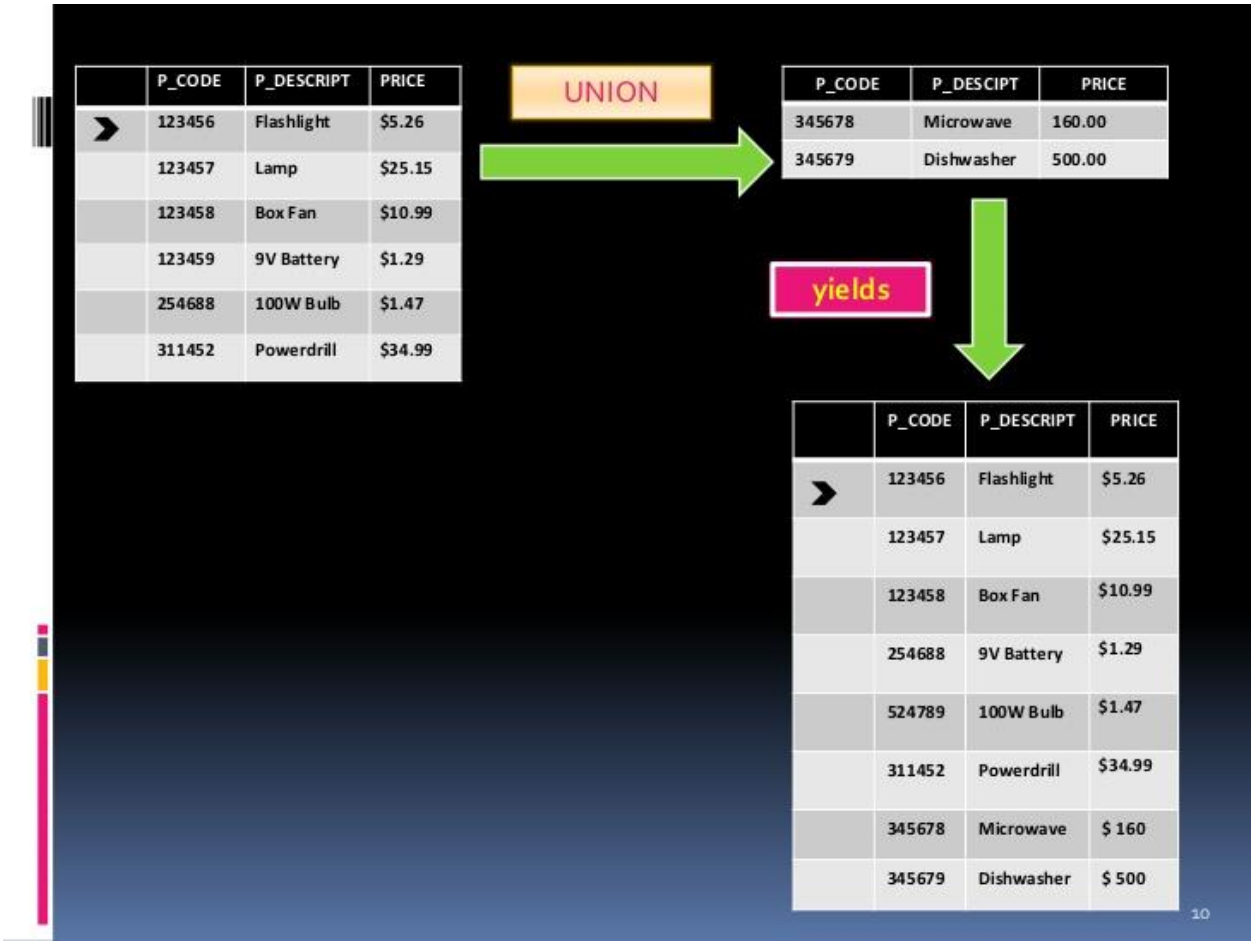
PROJECT

- Yields all values for selected attributes.
- It yields a vertical subset of a table.


ORIGINAL TABLE
NEW TABLE
P_CODE
P_DESCRIPT
PRICE
123456
Flashlight
$5.26
123457
Lamp
$25.15
123458
Box Fan
$10.99
123459
9V Battery
$1.29
254688
100W Bulb
$1.47
311452
Powerdrill
$34.99
PROJECT PRICE yields
PRICE
$5.26
$25.15
$10.99
$1.92
$1.47
$34.99
PROJECT P_DESCRIPT and PRICE yields
P_DESCRIPT
PRICE
Flashlight
$5.26
Lamp
$25.15
Box Fan
$10.99
9v battery
$1.92
100w bulb
$1.47
Powerdrill
$34.99
PROJECT P_CODE and PRICE yields
P_CODE
PRICE
123456
$5.26
123457
$25.15
123458
$10.99
213345
$1.92
254467
$1.47
311452
$34.99

UNION

- Combines all rows from two tables, excluding duplicate rows.
- To be used in UNION, tables must have the same attribute characteristics
- Columns, and domains must be compatible
- When two or more tables share the same number of columns, and when their corresponding columns share the same domains, they are said to be **UNION-COMPATIBLE**.


P_CODE
P_DESCRIPT
PRICE
123456
Flashlight
$5.26
123457
Lamp
$25.15
123458
Box Fan
$10.99
123459
9V Battery
$1.29
254688
100W Bulb
$1.47
311452
Powerdrill
$34.99
UNION
P_CODE
P_DESCIPT
PRICE
345678
Microwave
160.00
345679
Dishwasher
500.00
yields
P_CODE
P_DESCRIPT
PRICE
123456
Flashlight
$5.26
123457
Lamp
$25.15
123458
Box Fan
$10.99
254688
9V Battery
$1.29
524789
100W Bulb
$1.47
311452
Powerdrill
$34.99
345678
Microwave
$ 160
345679
Dishwasher
$ 500
10

INTERSECT

- Yields only the rows that appear in both tables.
- As was true in the case of UNION, the tables must be union-compatible to yield valid results.
- For example, you cannot use INTERSECT if one of the attributes is numeric and one is character-based.

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

INTERSECT

EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

yields

STU_FNAME	STU_LNAME
Franklin	Johnson

DIFFERENCE

- Yields all rows in one table that are not found in the other table;
- It subtracts one table form the other.
- Tables must be union-compatible to yield valid results.
- Note that subtracting the first table from the second table is not the same as subtracting the second table from the first table.

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

DIFFERENCE

EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

yields

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Martin	Lopez

PRODUCT

- Yields all possible pairs of rows from two tables
- Also known as Cartesian product
- Therefore, if one table has six rows and the other table has three rows,
- The PRODUCT yields a list composed of

6 x 3 = 18 rows

	P_CODE	P_DESCRIPT	PRICE
➤	123456	Flashlight	$5.26
	123457	Lamp	$25.15
	123458	Box Fan	$10.99
	123459	9V Battery	$1.29
	254688	100W Bulb	$1.47
	311452	Powerdrill	$34.99

PRODUCT

STORE	AISLE	SHELF
23	W	5
24	K	9
25	Z	6

yields

P_CODE	P_DESCRIPT	PRICE	STORE	AISLE	SHELF
123456	Flashlight	5.26	23	W	5
123456	Flashlight	5.26	24	K	9
123456	Flashlight	5.26	25	Z	6
123457	Lamp	25.15	23	W	5
123457	Lamp	25.15	24	K	9
123457	Lamp	25.15	25	Z	6
123458	Box Fan	10.99	23	W	5
123458	Box Fan	10.99	24	K	9
123458	Box Fan	10.99	25	Z	6
213345	9v battery	1.92	23	W	5
213345	9v battery	1.92	24	K	9
213345	9v battery	1.92	25	Z	6
311452	Powerdrill	34.99	23	W	5
311452	Powerdrill	34.99	24	K	9
311452	Powerdrill	34.99	25	Z	6
254467	100W bulb	1.47	23	W	5
254467	100W bulb	1.47	24	K	9
254467	100W bulb	1.47	25	Z	6

JOIN

- Allows information to be combined from two or more tables.
- It is the real power behind the relational database, allowing the use of independent tables linked by common attributes
- The CUSTOMER and AGENT tables shown will be used to illustrate several types of joins.

TABLE name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
1132445	Walker	32145	321
1217782	Adares	32145	125
1312243	Rakowski	34129	167
13212442	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

TABLE name: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

TABLE name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
1132445	Walker	32145	321
1217782	Adares	32145	125
1312243	Rakowski	34129	167
13212442	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

TABLE name: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445