SQL Date and DateTime Data Types

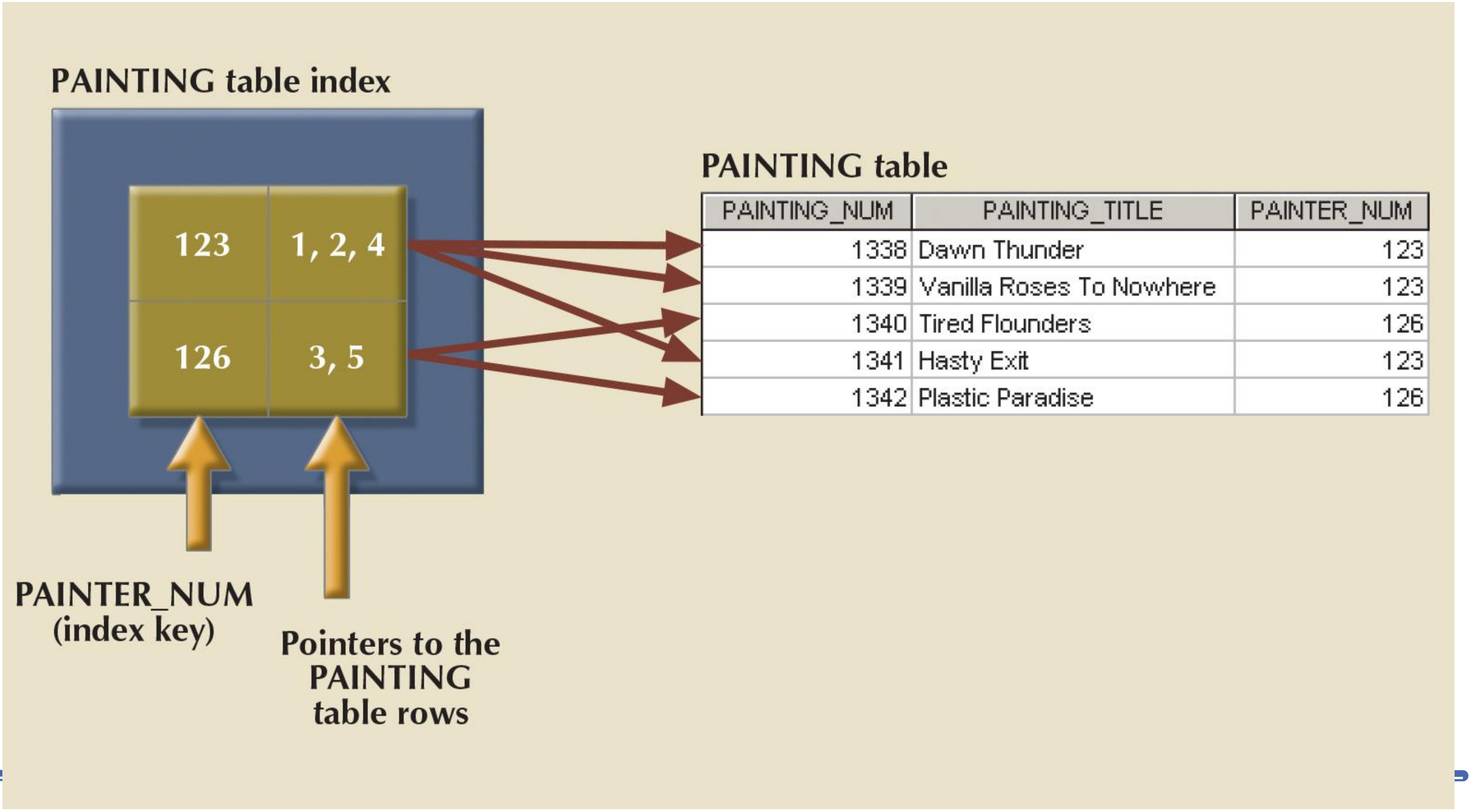
- The DATE type is used for values with a date part but no time part.
  - MySQL retrieves and displays DATE values in 'YYYY-MM-DD' format.
  - The supported range is '1000-01-01' to '9999-12-31'.
- The DATETIME type is used for values that contain both date and time parts.
  - MySQL retrieves and displays DATETIME values in 'YYYY-MM-DD HH:MM:SS' format.
  - The supported range is '1000-01-01 00:00:00' to '9999-12-31 23:59:59‘
- TIMESTAMP is used to record the date and time of an event
  - The time zone used is the server’s time zone
  - The supported range is '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC

Indexes

Indexes

- Indexes are created to provide quick access to data
  - Orderly arrangement to logically access rows in a table
- **Index key**: Index's reference point that leads to data location identified by the key
- **Unique index**: Index key can have only one pointer value associated with it
- Each index is associated with only one table
  - One table can have several indexes
  - Index is automatically created on the primary key column

Indexes



PAINTING_NUM	PAINTING_TITLE	PAINTER_NUM
1338	Dawn Thunder	123
1339	Vanilla Roses To Nowhere	123
1340	Tired Flounders	126
1341	Hasty Exit	123
1342	Plastic Paradise	126

SQL Indexes

- When primary key is declared, DBMS automatically creates unique index
- The **CREATE INDEX** command can be used to create indexes on the basis of any selected attribute
- **UNIQUE** qualifier prevents a value that has been used before
  - Composite indexes prevent data duplication
- To delete an index use the **DROP INDEX** command

SQL Indexes - Examples

Syntax:

CREATE [UNIQUE] INDEX *indexname*

ON *tablename* (*col1* [, *col2*]);

Examples:

```
CREATE UNIQUE INDEX P_CODEX
        ON PRODUCT (P_CODE);   -- Creates index on column P_CODE
CREATE INDEX PROD_PRICEX
        ON PROD (P_PRICE DESC);  -- Creates index in desc. order
DROP   INDEX   PROD_PRICEX;  -- Deletes index PROD_PRICEX
```

Modifying Table Structure

Modifying Table Structure

- **ALTER TABLE** command: To make changes in the table structure
- Keywords used with the command
  - ADD - Adds a column
  - MODIFY - Changes column characteristics
  - DROP - Deletes a column
- Also used to:
  - Add table constraints
  - Remove table constraints

Changing a Column’s Data Type and Data Characteristics

- ALTER used to change data type and characteristics
  - Some RDBMSs do not permit changes to data types unless column is empty
  - Changes in characteristics are permitted if they do not alter the existing data type
- Syntax:
  - Data Type: ALTER TABLE *tablename* MODIFY *(columnname(datatype))*;
  - Data Characteristic: ALTER TABLE *tablename* MODIFY *(columnname(characteristic))*;

Adding and Dropping Columns

- Adding a column
  - Use ALTER and ADD
  - Do not include the NOT NULL clause for new column
- Dropping a column
  - Use ALTER and DROP
  - Some RDBMSs impose restrictions on the deletion of an attribute

ALTER TABLE – Examples

This command adds a new column to the PRODUCT table

```
ALTER TABLE PRODUCT
      ADD (P_SALECODE CHAR(1));
```

This command modifies the column width

```
ALTER TABLE PRODUCT
      MODIFY  P_SALECODE CHAR(2);
```

This command deletes the column

```
ALTER TABLE PRODUCT
      DROP COLUMN P_SALECODE;
```

Deleting a Table from the Database

- **DROP TABLE**: Deletes table from database
  - Syntax - DROP TABLE *tablename*;
- Can drop a table only if it is not the one side of any relationship
  - RDBMS generates a foreign key integrity violation error message if you try to drop a referenced table

SQL’s Data Manipulation Language (DML)

Adding Data to a Table

- Add Table rows using the INSERT command

```
INSERT INTO tablename
        VALUES (value1, value2, ..., valueN);
```

- Example:

```
INSERT INTO VENDOR
        VALUES (21225, ‘Bryson, Inc.’, ‘Smithson’, ‘615’, ‘223-
        3234’, ‘TN’, ‘Y’);
```

- To view all data in the table, use the following command:

```
SELECT * FROM VENDOR
```

  - More on SELECT later

Adding Rows with Optional Attributes

- All NOT NULL columns need to be included in the INSERT command for adding a table row
- What to do when tables have several optional columns and no data needs to be added yet?
  - Use list of column names to specify what data is being entered
  - Example:

```
INSERT INTO PRODUCT (P_CODE, P_DESCRIPT)
      VALUES (‘BRT-345’, ‘Titanium drill bit’);
```

(Note: We are assuming here that only 2 columns are NOT NULL)

Saving Table Changes

- Changes not made permanent until saved in database
  - Power outage may result in loss of data
- Table contents can be saved by using the COMMIT command
- Syntax:
  START TRANSACTION (or BEGIN [WORK])

  COMMIT [WORK];
- COMMIT command permanently saves all changes made to any table in the database.

Restoring Table Contents

- Database can be restored to its previous condition using the ROLLBACK command
  - The changes should not have been permanently stored in the database through the COMMIT command
- Syntax:

```
ROLLBACK [WORK];
```

- COMMIT and ROLLBACK only work with the data manipulation commands that add, modify or delete table rows

Restoring Table Contents

- Example:
  1. CREATE a table called SALES
  2. INSERT 10 rows in the SALES table
  3. UPDATE 2 rows in the SALES table
  4. Execute the ROLLBACK command

- What does the ROLLBACK command do?
  - ROLLBACK will only undo the results of the INSERT and UPDATE commands

Deleting Table Rows

- Syntax:

```
DELETE FROM  tablename
     [WHERE       conditionlist];
```

- Examples:
  - To delete all data from the PRODUCT table

```
DELETE FROM PRODUCT;
```

  - To delete all rows with P_MIN = 5

```
DELETE FROM PRODUCT
   WHERE    P_MIN = 5;
```

Inserting Table Rows with SELECT

- Select subquery can be used to add multiple rows to a table, using another table as the source of data
  - Subquery is also called a nested query or inner query
- Syntax:

```
INSERT INTO    tablename
     SELECT    columnlist    FROM tablename;
```

  - Example:

```
INSERT INTO    TMP_PROD
     SELECT    P_CODE        FROM PRODUCTS;
```