Practical Aspects of Database Design

L4 - Database Session I

Stevens Institute of Technology

Disadvantages of Excel file

Purpose of Database System

Drawbacks of using file systems to store data

- Size of Data: When turns into a large amount of data, Spreadsheet solution will not work. It will takes long time to find a record from the multiple spreadsheet files.
- Ease of Updating Data: Multiple peoples cannot edit the same file on same time.
- Accuracy: The Data accuracy is hard to maintain and accuracy is in question.
- Security: You cannot secure the data in the text files and spreadsheet. Anyone can access the file and read any data present in the file.
- Redundancy: The duplication of data can be possible using text files or spreadsheet.
- Incomplete Data: Some of the data is important and needs to be validated

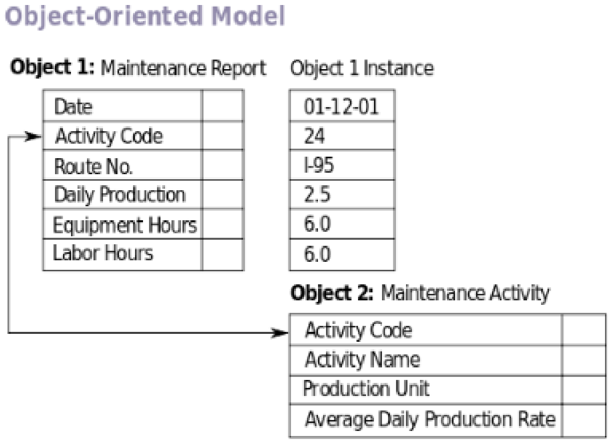
Database systems offer solutions to all the above problems

Object-relational Database Management System (ORDBMS)

- A database management system (DBMS) similar to a relational database, but with an object-oriented database model: objects, classes and inheritance are directly supported in database schemas and in the query language
- Allow attributes of tuples to have complex types, including nonatomic values such as nested relations.

Object-oriented Database

- A database management system in which information is represented in the form of objects as used in object-oriented programming



Object-Oriented Model

Object 1: Maintenance Report

Date	
Activity Code	
Route No.	
Daily Production	
Equipment Hours	
Labor Hours	

Object 1 Instance

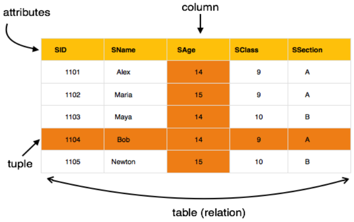
01-12-01
24
I-95
2.5
6.0
6.0

Object 2: Maintenance Activity

Activity Code	
Activity Name	
Production Unit	
Average Daily Production Rate	

Relational Database

- Tables: stores the relation among entities. Rows represents records and columns represent the attributes.
- Tuple: a single row of a table
- Relation instance: a finite set of tuples. Relation instances do not have duplicate tuples.
- Relation schema: describes the relation name (table name), attributes, and their names.
- Relation key: one or more attributes in each row to identify the row in the relation (table) uniquely.
- Attribute domain: Every attribute has some pre-defined value scope, known as attribute domain.



SID	SName	SAge	SClass	SSection
1101	Alex	14	9	A
1102	Maria	15	9	A
1103	Maya	14	10	B
1104	Bob	14	9	A
1105	Newton	15	10	B

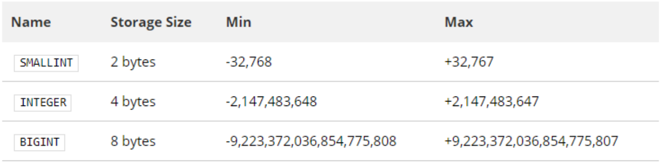
Introduction to SQL

What is SQL

- Virtually all relational database systems use SQL (Structured Query Language) for querying and maintaining the database.
- SQL is an ANSI (American National Standards Institute) standard

Domain Types in SQL

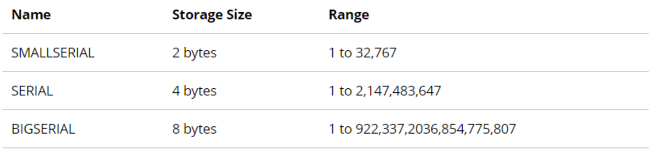
- boolean. Stores TRUE and FALSE values.
- char,varchar and text
  - char(n). Fixed userspecified length n.
  - varchar(n). Variable length with limit n.
  - text. Variable unlimited length
- numeric(p,d). Fixed point number, with userspecified precision of p digits, with d digits to the right of decimal point.
- integer



Name	Storage Size	Min	Max
`SMALLINT`	2 bytes	-32,768	+32,767
`INTEGER`	4 bytes	-2,147,483,648	+2,147,483,647
`BIGINT`	8 bytes	-9,223,372,036,854,775,808	+9,223,372,036,854,775,807

Domain Types in SQL

- real or double precision. Floating point or double-precision floating point numbers, with machine-dependent precision.
- float(n). Floating point number, with user-specified precision of at least n digits.
- serial. Create an auto-increment column using SERIAL pseudo type. A sequence is often used as a primary key.



Name	Storage Size	Range
SMALLSERIAL	2 bytes	1 to 32,767
SERIAL	4 bytes	1 to 2,147,483,647
BIGSERIAL	8 bytes	1 to 922,337,2036,854,775,807

Domain Types in SQL

- date.
  - It takes 4 bytes. The lowest and highest values of the DATE data type are 4713 BC and 5874897 AD.
  - When storing a date value, PostgreSQL uses the yyyy-mm-dd format e.g., 2000-12-31. It also uses this format for inserting data into a date column.
- time. Uses the TIME data type to manage time of day values. It requires 8 bytes and its allowed range is from 00:00:00 to 24:00:00.
- timestamp.
  - The timestamp data type allows you to store both date and time.
  - There are two temporal data types for handling timestamp, one without timezone (timestamp) and one with timezone (timestamptz).

SQL Rules

- Execute one query each time.
- Each query is ended with semicolon (;).
- SQL syntax is not case sensitive.
- Database names must be unique. Table names in one database must be unique. Column names in one table must be unique.
- Before using tables, you must get into the specific database.

Database

- Create database

```
CREATE DATABASE dbname;
```

- Drop database

```
DROP DATABASE [IF EXISTS] dbname;
```

Create Table

- create table

```
CREATE TABLE table_name (column_name1 column_type1, column_name2 column_type2,... )
```

- not null: can not contain NULL values

```
CREATE TABLE table_name (column_name1 column_type1 NOT NULL, ... )
```

Drop and Alter Table Constructs

- drop table: deletes all information about the dropped relation from the database.

```
DROP TABLE table_name
```

- alter table: command is used to add attributes to an existing relation
  - be used to add attributes to an existing relation:

```
ALTER TABLE table_name ADD col_name col_def;
```

  - be used to drop attributes of a relation

```
ALTER TABLE table_name DROP col_name col_def;
```

  - can also be used to alter attributes of a relation

```
ALTER TABLE table_name ALTER col_name col_def;
```

Insert Data

- Specify column name

```
INSERT INTO table_name (column1, column2...)
VALUES (value1, value2...);
```

- You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. However, make sure the order of the values is in the same order as the columns in the table.

```
INSERT INTO table_name VALUES (value1,value2...);
```

- Add multiple rows into a table at a time

```
INSERT INTO table (column1, column2, ...)
VALUES
(value1, value2, ...),
(value1, value2, ...) ,...;
```

Querying Data

- Select statement: selects data from a table.

```
SELECT expressions
FROM tables;
```

- Order By(optional): sorts the result set returned by the SELECT statement.

```
SELECT expressions
FROM tables
ORDER BY expression ASC | DESC
```

- Select Distinct(optional): removes duplicate rows in the result set.

```
SELECT DISTINCT expressions
FROM tables;
```

Filtering Data

- Where: filters rows based on a specified condition.

```
SELECT expressions
FROM tables
WHERE conditions
```

- Limit:
  - The statement returns n rows generated by the query.

```
SELECT expressions
FROM tables
LIMIT n;
```

  - In case you want to skip a number of rows before returning the n rows, you use OFFSET clause placed after the LIMIT clause

```
SELECT expressions
FROM tables
LIMIT n OFFSET m;
```

Filtering Data

- In: selects data that matches any value in a list of values.

```
SELECT expressions
FROM tables
WHERE value IN (val1, val2...) ;
```

- Between: selects data that is a range of values.

```
SELECT expressions
FROM tables
WHERE value BETWEEN low AND high ;
```

Filtering Data

- Like: filters data based on pattern matching.
  - Percent (%) for matching any sequence of characters.
  - Underscore (_) for matching any single character.

```
SELECT expressions
FROM tables
WHERE value LIKE val ;
```

Grouping Data

- The GROUP BY clause divides the rows returned from the SELECT statement into groups.
- For each group, you can apply an aggregate function e.g., SUM to calculate the sum of items or COUNT to get the number of items in the groups.
- The GROUP BY clause must appear right after the FROM or WHERE clause.
- Followed by the GROUP BY clause is one column or a list of comma-separated columns. You can also put an expression in the GROUP BY clause.

```
SELECT expressions
FROM tables
GROUP BY col_name ;
```

Grouping Data

- To filter groups, you use the HAVING clause instead of WHERE clause.
- You can use the HAVING clause without the GROUP BY clause. In this case, the HAVING clause will turn the query into a single group.
- HAVING cannot be placed before GROUP BY.

```
SELECT expressions
FROM tables
GROUP BY col_name
HAVING condition;
```

Summary: Order of Key Words

```
SELECT expressions
FROM tables
WHERE conditions
GROUP BY expressions
HAVING condition
ORDER BY expression ASC | DESC ;
```