SQL

Table of Contents

SQL

- **SQL**: structured query language used in relational databases
- DBMS and SQL support **CRUD** operations
  - **Create, read, update, delete**
- Wikipedia
- provides following capabilities
  - **Data definition language (DDL)**: define, set-up database
    * `CREATE`, `ALTER`, `DROP`
  - **Data manipulation language (DML)**: maintain, use database
    * `SELECT`, `INSERT`, `DELETE`, `UPDATE`
  - **Data control language (DCL)**: control access
    * `GRANT`, `REVOKE`
  - other commands: database administration, transaction control

Table creation

```
CREATE TABLE Account (
    AccountID smallint auto_increment, # surrogate key: DB auto-
       increments
    AccountName varchar(100), NOT NULL, # mandatory value
    OutstandingBalance DECIMAL(10, 2) NOT NULL,
    CustomerID smallint NOT NULL,
    AccountType enum('Personal', 'Company') NOT NULL, # enumerations
    PRIMARY KEY (CustomerID),    # specify primary key
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID) # specify
        foreign key
                    ON DELETE RESTRICT
                    ON UPDATE CASCADE
);
```

Insertion

- `"string"`
- `'enum'`
- `""` is different to `NULL`
- with columns specified

```
INSERT INTO Customer
    (CustFirstName, CustLastName, CustType)
    VALUES ("Peter", "Smith", 'Personal');
```

- if columns are not specified, you must enter all columns

```
INSERT INTO Customer
    (CustFirstName, CustLastName, CustType)
    VALUES (DEFAULT, "James", NULL, "Jones", "JJ Enterprises", 'Company
        ');
```

Selection

MySQL style SELECT selected keywords

`SELECT [ALL | DISTINCT] select_expr [, select_expr ...]` - List the columns (and expressions) that are returned from the query `[FROM table_references]` - Indicate the table(s) or view(s) from where the data is obtained - `ColName AS NewColName`: rename columns

`[WHERE where_condition]` - Indicate the conditions on whether a particular row will be in the result - `[LIKE "<regex>"]` - used for finding records that match a pattern - `%`: 0+ characters - `_`: single character - e.g. `WHERE CustomerName LIKE "a%"` finds values starting with a

`[GROUP BY col_name | expr } [ASC | DESC], ...]` - Indicate categorisation of results

`[HAVING where_condition ]` - Indicate the conditions under which a particular category (group) is included in

`[ORDER BY col_name | expr | position } [ASC | DESC], ...]` - Sort the result based on the criteria - Default is `ASC`

`[LIMIT offset ,] row_count | row_count OFFSET offset}]` - Limit which rows are returned by their return order (ie 5 rows, 5 rows from row 2) - `LIMIT n`: limits output size - `OFFSET x`: skips first `x` records

Aggregation

- operate on subset of values in a column of a relation (table), returning a single value
- allows you to produce derived attributes
- e.g. `AVG(), COUNT(), MIN(), MAX(), SUM()`
    - all of these (except `COUNT()`) return the result ignoring `NULL` values
    - `COUNT()` counts the number of records
- MySQL GroupBy Functions

e.g. count customers

```sql
SELECT COUNT(CustomerID)
FROM Customer;
```

e.g. average balance per customer

```sql
SELECT AVG(OutstandingBalance)
FROM Account
GROUP BY CustomerID;
```

Group by, having

- **group by** groups records over a set of attributes
    - often used with aggregation
    - to put a selection condition over a group by statement, use a `HAVING` clause
- e.g. average balance per customer, for customers whose average balance is over 10000

```sql
SELECT AVG(OutstandingBalance)
FROM Account
GROUP BY CustomerID
HAVING AVG(OutstandingBalance) > 10000
```

Joins

- Cross product: not very useful

```sql
SELECT * FROM Rel1, Rel2
```

- Inner/equi join: joins tables over keys using specified condition

```sql
SELECT * FROM Customer INNER JOIN Account
    ON Customer.CustomerID = Account.CustomerID;
```

- Natual join: joins tables over keys; you don't need to specify condition, but key attributes must have identical name

```sql
SELECT * FROM Customer NATURAL JOIN Account;
```

- Outer Join: joins tables over keys; left/right, including records that don't match the join from the other table

```sql
SELECT * FROM Customer LEFT OUTER JOIN Account
    ON Customer.CustomerID = Account.CustomerID;
```