# SQL

IDATG2204 Data Modelling and Database Systems

Norwegian University of Science and Technology

# Where are We Now?

- W02: Introduction, Relational Algebra
- W03: SQL
- W04: SQL, Conceptual Modelling
- W05: Conceptual Modelling
- W06: Normalisation
- W07: Logical Modelling, NOSQL
- W08: DB Application Development
- W09: DB Security, Project Kick-off
- W10-W14: Project Work with Peer Review
- W15: Indexing, query processing, concurrency
- W16: Recovery
- W17: More SQL and NOSQL
- W18: Review and Wrap-up

# Outline

- Intro to SQL
- SQL SELECT:
  - SELECT, DISTINCT, ORDER BY
  - FROM
  - WHERE
  - Subqueries
  - GROUP BY
    - HAVING
  - Union

# Intro to SQL

- SQL is a **declarative** database query language:
  - Describing what the result would look like
  - Leave to the DBMS to decide how to create the result
- SQL reserved words are typically written in upper case
- Database languages are divided in two sub-languages:
  - Data Manipulation Language (DML)
    - For modifying user data: Create, Retrieve, Update, Delete (CRUD)
    - `SELECT`, `INSERT`, `UPDATE`, `DELETE`
  - Data Definition Language (DDL)
    - For modifying database catalogue/data dictionary
    - `CREATE`, `ALTER`, `DROP`

# Intro to SQL SELECT

- General form:

```
SELECT    [DISTINCT] {* | [columnExpression [alias]]
                      [, …]}
FROM      TableName [alias] [[, …] | [joinExpression]]
[WHERE    condition]
[GROUP BY columnList] [HAVING condition]
[ORDER BY columnList]
```

- Basic sequence of processing:

```
1.  FROM
2.  WHERE
3.  GROUP BY
4.  HAVING
5.  SELECT
6.  ORDER BY
```

# Outline

- Intro to SQL
- SQL SELECT:
  - SELECT, DISTINCT, ORDER BY
  - FROM
  - WHERE
  - Subqueries
  - GROUP BY
    - HAVING
  - Union

# SQL SELECT (1)

- General form:
  - `SELECT [DISTINCT] {* | [columnExpression [alias]] [, …]}`
  - What relational algebra operation is this equal to?
    - The projection operation (++): $\Pi_{a1, ..., an}(R)$
- Wildcard `(*)`:
  - For retrieving all columns in the relation
  - Example: Find all cars for sale:
    - `SELECT * FROM cars`
- `columnExpression` may be a list of columns:
  - Example: Find the makes and the models of cars for sale:
    - `SELECT make, model FROM car`

# SQL SELECT (2)

- General form:
  - `SELECT [DISTINCT] {* | [columnExpression [alias]] [, …]}`
- `columnExpression` may also be:
  - Calculated fields, e.g.:
    - `SELECT CONCAT(UPPER(make), ' ', UPPER(model))`
    - `SELECT MIN(mileage)`
- `alias`:
  - For renaming some of the attributes in the result relation
    - `SELECT MIN(mileage) AS smallest FROM car`
  - What relational algebra operation is this equal to?
    - The rename operation: $\rho_{S(a1, ..., an)}(R)$

# SQL SELECT (3)

- General form:
  - `SELECT [DISTINCT] {* | [columnExpression [alias]] [, …]}`
- `DISTINCT`:
  - When we want duplicates removed, e.g.,:
    - `SELECT DISTINCT year FROM car`
- Example:
  - List unique car and model names, renaming columns to Norwegian:
    - `SELECT DISTINCT make AS merke, model AS modell`

# ORDER BY

- Data is to be considered unordered in an RDBMS:
  - The RDBMS decides on the order
- We may ask the RDBMS to return tuples in a specific order:
  - `ORDER BY column [DESC] [, …]`
- Example
  - Order car tuples on make, model, and year:
    - `ORDER BY make, model, model_year`
  - Order car tuples on year (descending), make, model:
    - `ORDER BY model_year DESC, make, model`

# Outline

- Intro to SQL
- SQL SELECT:
  - SELECT, DISTINCT, ORDER BY
  - FROM
  - WHERE
  - Subqueries
  - GROUP BY
    - HAVING
  - Union

# SQL `FROM`

- The FROM clause is where we define what table to query:
  - `SELECT * FROM car`
- or what tables to combine data from and how to join:
  - INNER JOIN - for equijoins in the relational algebra:
    $R \bowtie_F S$
  - LEFT/RIGHT OUTER JOIN - as for the relational algebra:
    R ⟕F S
    R ⟖F S

# SQL `INNER JOIN`

- Example:
  - Find car information together with dealer city and county name:

    -
      ```
      SELECT car.*, city, name FROM car
      INNER JOIN dealer ON dealer_id = dealer.id
      INNER JOIN county ON county_no = no
      ```

# SQL OUTER JOIN

- Example:
  - Find car information along dealer city and county name, for all cities and counties even when there are no cars:

    -
      ```
      SELECT car.*, city, name FROM car
      RIGHT OUTER JOIN dealer ON dealer_id = dealer.id
      RIGHT OUTER JOIN county ON county_no = no
      ```

# Outline

- Intro to SQL
- SQL SELECT:
  - SELECT, DISTINCT, ORDER BY
  - FROM
  - WHERE
  - Subqueries
  - GROUP BY
    - HAVING
  - Union

# SQL `WHERE` (1)

- What relational algebra operation is this equal to?
  - The selection operation: $\sigma_F(R)$
  - Five basic search conditions/predicates (repeated):
    - Comparison (=, <>, <, …)
    - Range (`[NOT] BETWEEN`)
    - Set membership (`[NOT] IN (…)]`)
    - Pattern match (`[NOT] LIKE '…']`)
    - Null (`IS [NOT] NULL`)
  - Boolean expressions (repeated):
    - `AND`, `OR`, `NOT`, `(`, `)`

# SQL WHERE (2)

- MariaDB has many built-in functions:
  - https://mariadb.com/kb/en/built-in-functions/
- Example: Find cars from 2017-2019 where mileage is no more than 40000 and fuel type is not electric or hybrid:

  -
    ```
    SELECT * FROM car
    WHERE model_year BETWEEN 2017 AND 2019
      AND mileage <= 40000
      AND fuel NOT IN ('electric', 'hybrid')
    ```

# Outline

- Intro to SQL
- SQL SELECT:
  - SELECT, DISTINCT, ORDER BY
  - FROM
  - WHERE
  - Subqueries
  - GROUP BY
    - HAVING
  - Union

# Introduction to Subqueries

- Types of subqueries:
  - Scalar subqueries:
    - Returning scalar values,
      e.g.: `SELECT AVG(price) FROM car`
  - (Row subquery)
  - Table subqueries:
    - Returning a table,
      e.g., `SELECT DISTINCT county_no FROM dealer`
  - Boolean subqueries:
    - `EXISTS/NOT EXISTS`

# Aggregation Subquery

- Example:
  - Find all cars of the oldest model:

    -
      ```
      SELECT * FROM car
      WHERE model_year =
                      (SELECT MIN(model_year) FROM car)
      ```
  - List the make, model, and year of all cars that are more expensive than the average car price, and show by how much the price is greater than the average:

    -
      ```
      SELECT make, model, model_year, price -
                     (SELECT AVG(price) FROM car)
      FROM car
      WHERE price > (SELECT AVG(price) FROM car)
      ```

# ALL Subquery

- The ALL keyword may be used with subqueries that produce a single column of numbers
  - The condition is only true if it is satisfied by all values produced
- Example:
  - List the model years for which there are fewest cars for sale:

    -
      ```
      SELECT model_year, COUNT(id) as count
      FROM car
      GROUP BY model_year
      HAVING count <= ALL
        (SELECT COUNT(id)
          FROM car
          GROUP BY model_year)
      ```

# IN/NOT IN Subquery

- We have already used IN/NOT IN:
  - `WHERE fuel NOT IN ('hybrid', 'electric')`
- A similar set may be produced by a subquery
- IN subqueries should usually be replaced by JOIN
- Example:
  - Find the names of counties where there are no dealers:

    -
      ```
      SELECT name
      FROM county
      WHERE no NOT IN
          (SELECT DISTINCT county_no
           FROM dealer)
      ```

# (Row Subquery Example)

- Say we are interested in any Volkswagen Passat, Toyota Corolla, or Audi A3:

  -
    ```
    SELECT * FROM car
    WHERE (make, model) IN
          (('Volkswagen', 'Passat'),
           ('Toyota', 'Corolla'),
           ('Audi', 'A3'))
    ```
- If the user preferences are stored in table `user_pref`:

  -
    ```
    SELECT * FROM car
    WHERE (make, model) IN
       (SELECT make, model
        FROM user_pref
        WHERE user_id = 'rune.Hjelsvold@ntnu.no')
    ```

# `EXISTS/NOT EXISTS` Subquery

- Produce true or false depending on whether the subquery is non-empty or empty, respectively
- The subquery "sees" attributes in the outer query
- Is usually more efficient than equivalent `IN/NOT IN` subquery
- Example:
  - Find the names of counties where there are no dealers:

    -
      ```
      SELECT name
      FROM county
      WHERE NOT EXISTS
          (SELECT *
           FROM dealer
           WHERE no = county_no)
      ```

# Outline

- Intro to SQL
- SQL SELECT:
  - SELECT, DISTINCT, ORDER BY
  - FROM
  - WHERE
  - Subqueries
  - GROUP BY
    - HAVING
  - Union

# SQL GROUP BY (1)

- What relational algebra operation is this equal to?
  - The grouping operation: $_{GA}\ _{AL}$(R)
- The attributes returned in a group by query may only be:
  - Grouping attributes
  - Attributes with only one value per group
  - Aggregate values
- Example:
  - Find the min, max, and average price of cars per dealer (city):

    -
      ```
      SELECT city, MIN price, MAX price, AVG price
      FROM car
           INNER JOIN dealer ON dealer_id = dealer.id
      GROUP BY dealer.id
      ```

# SQL `GROUP BY` (2)

- Aggregation variations:
  - Aggregating distinct values only:
    - `COUNT(DISTINCT model_year)`
  - Counting number of defined values:
    - `COUNT(comment)`
  - Counting number of tuples:
    - `COUNT(*)`

# SQL `HAVING`

- The `WHERE` clause identifies tuples to be grouped:
  - $_{GA\ \ AL}(\sigma_{F}(R))$
- The `HAVING` clause identifies what groups to return:
  - $\sigma_{F\_AL}(_{GA\ \ AL}(R))$
- Example:
  - Find the number of cars of each make and model, but only when there are more than 2 cars of that make and model:

    -
      ```
      SELECT make, model, COUNT(id) AS car_count
      FROM car
      GROUP BY make, model
      HAVING car_count > 2
      ```

# Outline

- Intro to SQL
- SQL SELECT:
  - SELECT, DISTINCT, ORDER BY
  - FROM
  - WHERE
  - Subqueries
  - GROUP BY
    - HAVING
  - Union

# SQL `UNION`

- The `UNION` can be used to merge different query results:
  - `R ∪ S`
- Example:
  - Find the names of counties, dealer cities, and car makers:

    -
      ```
      SELECT name FROM county
      UNION
      SELECT city FROM dealer
      UNION
      SELECT make FROM car
      ```