# DM534: Introduction to Relational Databases

# (Part 2)

**2019**

**Slides by Christian Wiwie**
**(edits by Rolf Fagerberg)**

SDU
UNIVERSITY OF
SOUTHERN DENMARK

# Relational Query Languages

- Based on relational algebra
- For relational databases, i.e. relational data model
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic
  - Allows for much optimization
- **SQL** (Structured Query Language)
  - Most widely used relational query language

→ Understanding Relational Algebra is key to understanding SQL, query processing!

# What is an “Algebra”?

- Mathematical system consisting of
  - **Operands:** Values from which new values can be constructed by applying operations
  - **Operations:** Procedures that construct new values from given values
  - **Operators:** Symbols denoting operations
- Variables are letters that can represent values

# Examples for Algebras

- Integer algebra
  - **Operands**: The set of integers [..., -1, 0, 1, ...]
  - **Operations**: Addition, subtraction, multiplication, division, ...
  - **Operators**: +, -, *, /, ...
- Example for algebraic expressions:

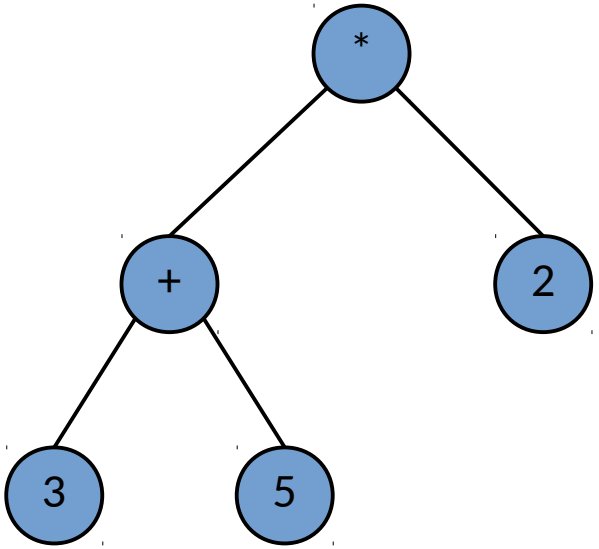
  (3 + 5) * 2

  5 – x / 3

# Algebraic expressions

- Can be visualized as expression trees

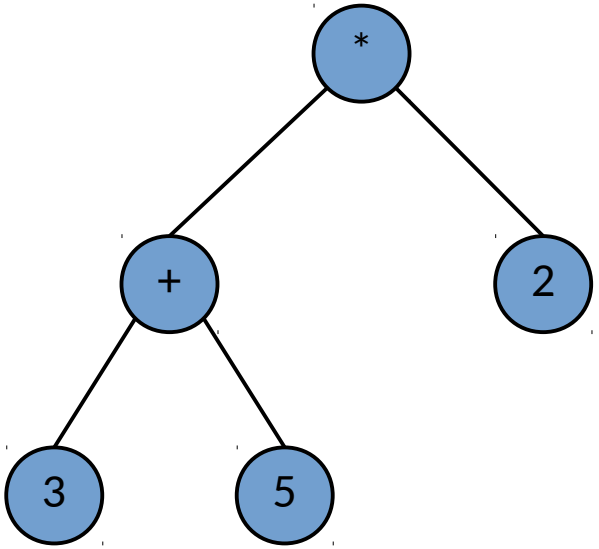
  (3 + 5) * 2

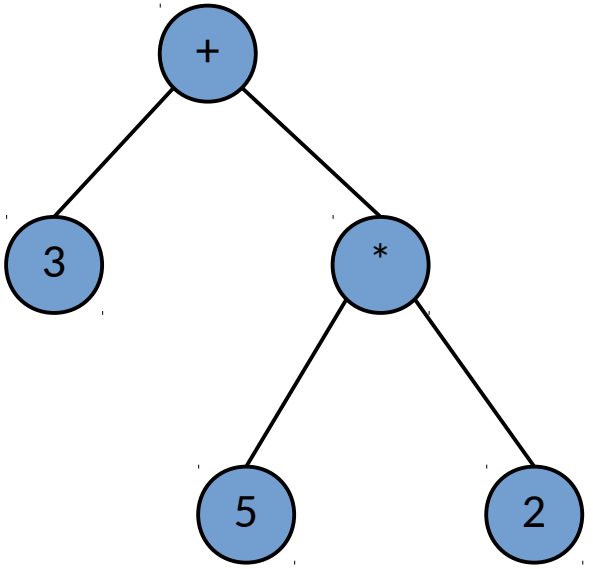


# Algebraic expressions

- Can be visualized as expression trees

(3 + 5) * 2 vs. 3 + 5 * 2

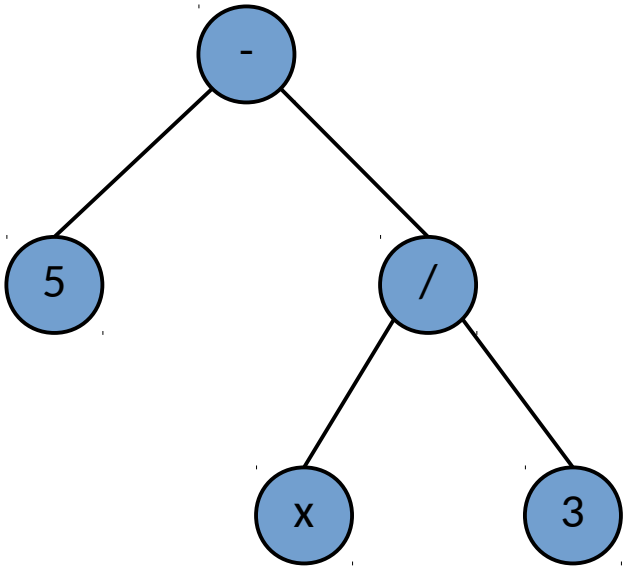


# Algebraic expressions

- Can be visualized as expression trees

  5 – x / 3

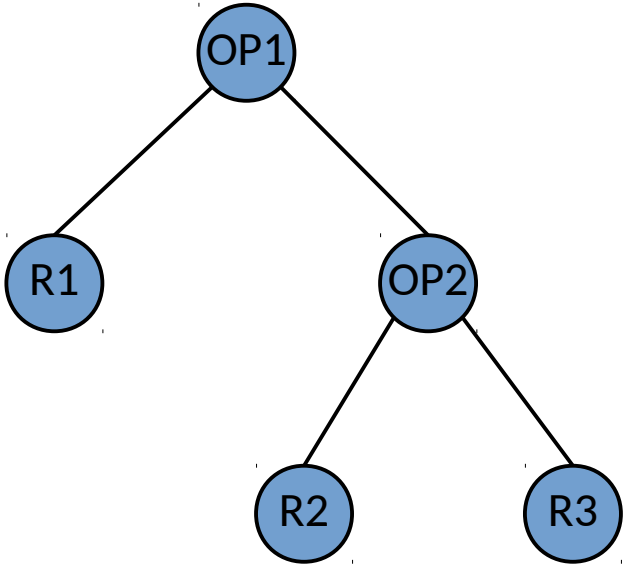


# What is Relational Algebra?

- An algebra where
  - **operands** are relations
  - **operations** compute new relations from relations
- Can be used as a query language for relations
  - "Language" of relational databases

# What is Relational Algebra?

=> Expressions of relational algebra can also be visualized as trees




- OP1 and OP2 are relational operations
- R1, R2, R3 are variables for relations

# Relational Algebra: 5 Basic Operations

- Selection: $\sigma_C(R)$

  Selects a subset of tuples from relation R, for which condition C holds (horizontal)
- Projection: $\pi_{A_1,\ldots,A_k}(R)$

  Retains attributes $A_1, \ldots, A_k$ from relation R (vertical)
- Cross-product: **R1 x R2**

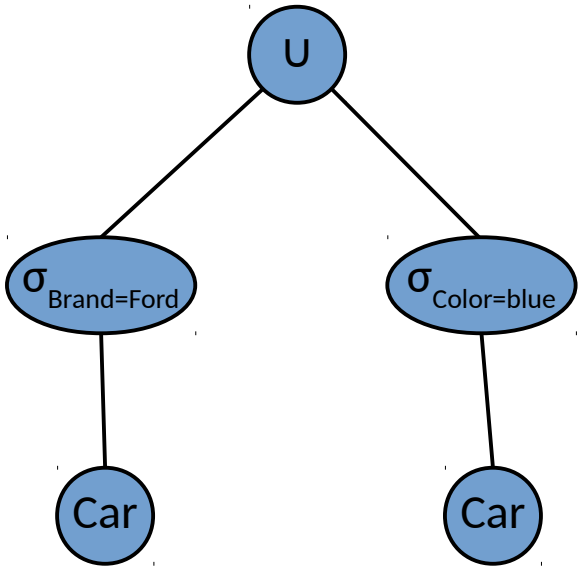
  Pairwise combination of tuples of relations R1 and R2
- Set-difference: **R1 – R2**

  Tuples in relation R1, but not in relation R2
- Union: **R1 ∪ R2**

  Tuples in relation R1 and/or in relation R2
- Since each operation returns a relation, operations can be composed (Algebra is “closed”)

# What is Relational Algebra?

=> An expression tree could like this




- What does this express?

# Relational Algebra: Example Instances

## Example Instances

**Sailing Database:**
**Sailors, Boats, Reserves**

***Boats:***

| bid | bname | color |
|---|---|---|
| 101 | Interlake | blue |
| 102 | Interlake | red |
| 103 | Clipper | green |
| 104 | Marine | red |

***Reserves1:***

| sid | bid | day |
|---|---|---|
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

***Sailers1:***

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

***Sailers2:***

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

DM505 Database Design and Programming

# Selection (σ)

**Selects rows that satisfy *selection condition*.**
**Result is a relation.**

*Schema* of result is same as that of the input relation.

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

$S2$

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 58 | rusty | 10 | 35.0 |

$$\sigma_{rating>8}(S2)$$

# Projection (π)

**Examples:** $\pi_{age}(S2)$; $\pi_{sname,rating}(S2)$

**Retains only attributes that are in the *"projection list"*.**

***Schema* of result:**

- exactly the fields in the projection list,
- with the same names that they had in the input relation.

**Projection operator has to *eliminate duplicates* (How do they arise? Why remove them?)**

- Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it. (Why not?)

# Projection (π)

**Projection**

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

**S2**

| sname | rating |
|---|---|
| yuppy | 9 |
| lubber | 8 |
| guppy | 5 |
| rusty | 10 |

$\pi_{sname,rating}(S2)$

| age |
|---|
| 35.0 |
| 55.5 |

$\pi_{age}(S2)$

# Cross Product

**S1 x R1: Each row of S1 paired with each row of R1.**

**Q: How many rows in the result?**

***Result schema* has one field per field of S1 and R1, with field names `inherited' if possible.**

- *May have a naming conflict*: Both S1 and R1 have a field with the same name.
- In this case, can use the *renaming operator*:

$$\rho\,(C(1 \to sid1, 5 \to sid2),\ S1 \times R1)$$

# Cross Product

S1

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

R1

| sid | bid | day |
|---|---|---|
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

$$\rho(C(1 \rightarrow sid1, 5 \rightarrow sid2), S1 \times R1) =$$

| sid1 | sname | rating | age | sid2 | bid | day |
|---|---|---|---|---|---|---|
| 22 | dustin | 7 | 45.0 | 22 | 101 | 10/10/96 |
| 22 | dustin | 7 | 45.0 | 58 | 103 | 11/12/96 |
| 31 | lubber | 8 | 55.5 | 22 | 101 | 10/10/96 |
| 31 | lubber | 8 | 55.5 | 58 | 103 | 11/12/96 |
| 58 | rusty | 10 | 35.0 | 22 | 101 | 10/10/96 |
| 58 | rusty | 10 | 35.0 | 58 | 103 | 11/12/96 |

# Union and Set Difference

**All of these operations take two input relations, which must be *union-compatible*:**

- Same number of fields.
- `Corresponding' fields have the same type.

**For which, if any, is duplicate elimination required?**

# Union

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

**S1**

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

**S2**

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |
| 44 | guppy | 5 | 35.0 |
| 28 | yuppy | 9 | 35.0 |

$S1 \cup S2$

# Set Difference

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

***S1***

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

***S2***

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |

***S1 – S2***

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 44 | guppy | 5 | 35.0 |

***S2 – S1***

# Nesting Operators

- Result of a relational algebra operator is a relation
- It can be used as input to another relational algebra operator

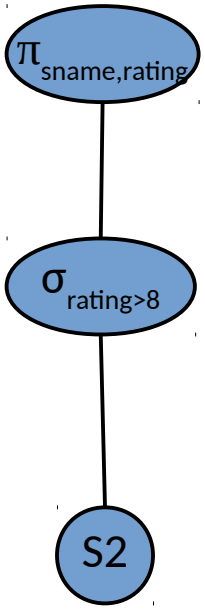
| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

| sname | rating |
|---|---|
| yuppy | 9 |
| rusty | 10 |

$$\pi_{sname,rating}(\sigma_{rating>8}(S2))$$

# Nesting Operators

- As expression tree:



# Compound Operator: Intersection

- In addition to the 5 basic operators, there are several additional “Compound Operators”
  - Do not add computational power to the language
  - Useful shorthands
  - Can be expressed with basic operations
- Example: **Intersection**
  - Takes two input relations that are union-compatible

$$R \cap S = R - (R - S)$$

# Compound Operator: Intersection

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

**S1**

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

**S2**

| sid | sname | rating | age |
|---|---|---|---|
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

**S1 ∩ S2**

# SQL - A language for Relational DBs

# SQL - A language for Relational DBs

- Say: “ess-cue-ell” or “sequel”
  - But spelled “SQL”
- Data Definition Language (DDL)
  - create, modify, delete relations
  - specify constraints
  - administer users, security, etc.
- Data Manipulation Language (DML)
  - Specify queries to find tuples that satisfy criteria
  - add, modify, remove tuples
- The DBMS is responsible for efficient evaluation

# SQL - A language for Relational DBs

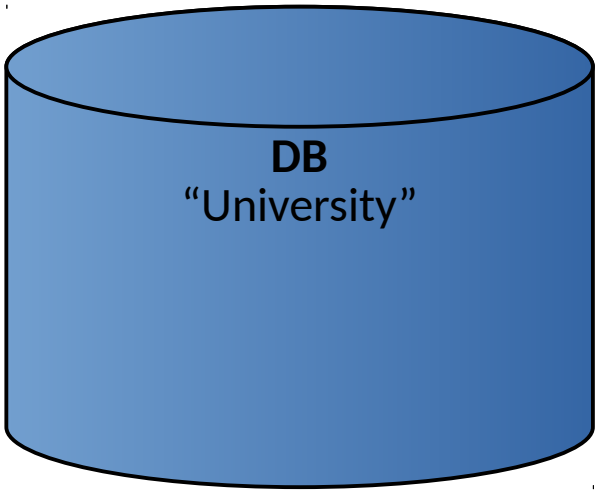
- Query language to retrieve data from database
- Includes a data-definition component to define database schemas
- SQL commands have to be terminated with ‘;’
- SQL is standardized
  - some DBMS include their own SQL commands

# Creating Databases in SQL

- Create a new, empty database ‘University’:

  CREATE DATABASE University;

  – Does not contain any relations upon creation

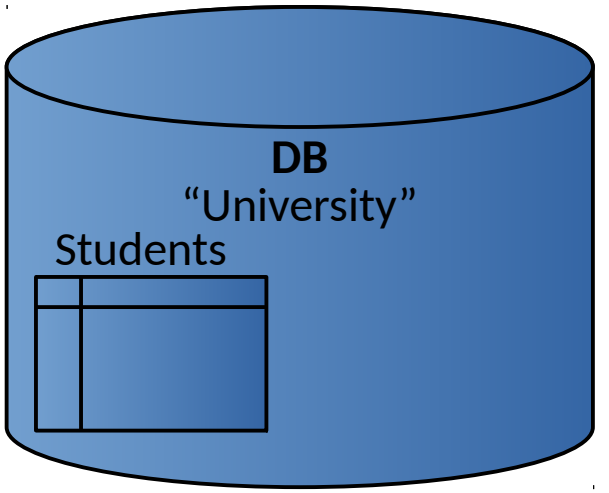


# Creating Relations in SQL

- Create a new, empty relation ‘Students’:

  CREATE TABLE Students (sid CHAR(20) PRIMARY KEY, name CHAR(20), login CHAR(10), age INTEGER, gpa FLOAT);

  - Does not contain any tuples upon creation
  - Note: the type (domain) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.

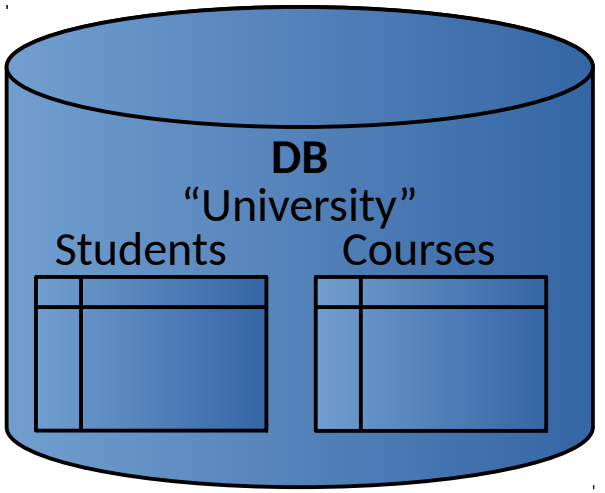



| sid | name | login | age | gpa |
|---|---|---|---|---|

# Creating Relations in SQL

- Similarly:

```
CREATE TABLE Courses
( cid CHAR(20) PRIMARY KEY, cname CHAR(20), credits
INTEGER);
```



# Adding and Deleting Tuples

- Insert a single tuple:

```
INSERT INTO Students (sid, name, login, age, gpa)
VALUES ('53688', 'Smith', 'smith@ee', 18, 3.2);
```

| sid | name | login | age | gpa |
|---|---|---|---|---|
| 53688 | Smith | smith@ee | 18 | 3.2 |

- Delete all tuples satisfying some condition (e.g., name = Smith):

```
DELETE FROM Students S WHERE S.name = 'Smith';
```

| sid | name | login | age | gpa |
|---|---|---|---|---|

# Selecting Tuples in SQL

- Find tuples for all 18 year old students with gpa's above 2.0:

  SELECT * FROM Students S WHERE S.age=18 AND S.gpa > 2.0;

| sid | name | login | age | gpa |
|---|---|---|---|---|
| 53688 | Smith | smith@ee | 18 | 3.2 |

- To get just names and logins:

  SELECT S.name, S.login FROM Students S WHERE S.age=18 AND S.gpa > 2.0;

| name | login |
|---|---|
| Smith | smith@ee |

# Relational Algebra Operators in SQL

- Relational algebra operators can be expressed with SQL
- Selection operator (σ):

```
SELECT * FROM Students S
    WHERE S.age=18 AND S.gpa > 2.0;
```

- Projection operator (π):

```
SELECT S.age,S.gpa FROM Students S;
```

- Union:

```
SELECT * FROM Students S
    WHERE S.age=18 AND S.gpa > 2.0
UNION
SELECT * FROM Students S
    WHERE S.age=20 AND S.gpa > 2.3;
```

# Relational Algebra Operators in SQL

- Set Difference:

```
SELECT * FROM Students S
    WHERE S.gpa > 2.0
EXCEPT
SELECT * FROM Students S
    WHERE S.age=19;
```

- Cross Product:

```
SELECT * FROM Students S, Enrolled E;
```

# Primary Keys in SQL

- Single attribute primary key:

  CREATE TABLE Students (sid CHAR(20) PRIMARY KEY, name CHAR(20), login CHAR(10), age INTEGER, gpa FLOAT)

- Multi-attribute primary key:

  CREATE TABLE Enrolled (sid CHAR(20) cid CHAR(20), grade CHAR(2), PRIMARY KEY (sid,cid))

# Foreign Keys in SQL

- Only students listed in the Students relation should be allowed to enroll for courses

  → sid is a foreign key referring to Students

- Students can only enroll for registered courses

  → cid is a foreign key referring to Courses

```
CREATE TABLE Enrolled
    (sid CHAR(20),cid CHAR(20),grade CHAR(2),
    PRIMARY KEY (sid,cid),
    FOREIGN KEY (sid) REFERENCES Students,
    FOREIGN KEY (cid) REFERENCES Courses);
```

# Thank you for your attention!