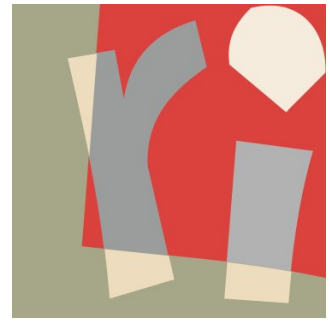

Uncertain Data Management
Reminders on Relational Algebra and Calculus

Antoine Amarilli[1], Silviu Maniu[2]

November 21st, 2017

[1]Télécom ParisTech

[2]LRI

Table of contents

Relations

- Relation name and arity
- Attribute names (optional)

Relation Class, arity 5

date	teacher	resp	name	num

Relations

- Relation name and arity
- Attribute names (optional)

Relation Class, arity 5

date	teacher	resp	name	num
2017-11-21	Antoine	Fabian	Uncert. Data Mgmt	1
2017-11-28	Antoine	Fabian	Uncert. Data Mgmt	2
2017-12-05	Antoine	Fabian	Uncert. Data Mgmt	3
2017-12-12	Silviu	Fabian	Uncert. Data Mgmt	4

- Set of rows (tuples) on a domain (no duplicates)

Relational signature

- Signature σ: set of relation names and attributes, e.g.:
  - Class(**date**, **teacher**, **resp**, **name**, **num**)
  - Student(**id**, **name**)
  - Member(**student**, **classname**)
- Database instance: one relation for each name in σ

Relational signature

- Signature σ: set of relation names and attributes, e.g.:
  - Class(**date**, **teacher**, **resp**, **name**, **num**)
  - Student(**id**, **name**)
  - Member(**student**, **classname**)
- Database instance: one relation for each name in σ

$\rightarrow$ Query:

- Input: database
- Output: relation

Languages

Two languages to write queries:

- The relational algebra:
  - operational way to define queries
  - based on operators to construct new relations
- The relational calculus:
  - declarative way to define queries
  - based on first-order logic

Languages

Two languages to write queries:

- The relational algebra:
  - operational way to define queries
  - based on operators to construct new relations
- The relational calculus:
  - declarative way to define queries
  - based on first-order logic

→ Codd's theorem: both have the same expressive power

Languages

~~Two~~ Three languages to write queries:

- The relational algebra:
  - operational way to define queries
  - based on operators to construct new relations
- The relational calculus:
  - declarative way to define queries
  - based on first-order logic

→ Codd's theorem: both have the same expressive power

- SQL, the practical language used by databases

Table of contents

Relational algebra

- Basic relations:
  - the relation names in the signature
  - constant relations, e.g., the empty relation
- Projection Π
- Selection σ
- Renaming ρ
- Union $\cup$
- Product $\times$ and join $\bowtie$
- Difference $-$

Projection: keep a subsequence of the attributes

Class

date	teacher	resp	name	num
2017-11-21	Antoine	Fabian	Uncert. Data Mgmt	1
2017-11-28	Antoine	Fabian	Uncert. Data Mgmt	2
2017-12-05	Antoine	Fabian	Uncert. Data Mgmt	3
2017-12-12	Silviu	Fabian	Uncert. Data Mgmt	4

Projection: keep a subsequence of the attributes

Class

date	teacher	resp	name	num
2017-11-21	Antoine	Fabian	Uncert. Data Mgmt	1
2017-11-28	Antoine	Fabian	Uncert. Data Mgmt	2
2017-12-05	Antoine	Fabian	Uncert. Data Mgmt	3
2017-12-12	Silviu	Fabian	Uncert. Data Mgmt	4

$\Pi_{\textbf{teacher},\textbf{resp}}(\text{Class})$

teacher	resp

Projection: keep a subsequence of the attributes

Class

date	teacher	resp	name	num
2017-11-21	Antoine	Fabian	Uncert. Data Mgmt	1
2017-11-28	Antoine	Fabian	Uncert. Data Mgmt	2
2017-12-05	Antoine	Fabian	Uncert. Data Mgmt	3
2017-12-12	Silviu	Fabian	Uncert. Data Mgmt	4

$\Pi_{\textbf{teacher},\textbf{resp}}(\text{Class})$

teacher	resp
Antoine	Fabian
Silviu	Fabian

Projection: keep a subsequence of the attributes

Class

date	teacher	resp	name	num
2017-11-21	Antoine	Fabian	Uncert. Data Mgmt	1
2017-11-28	Antoine	Fabian	Uncert. Data Mgmt	2
2017-12-05	Antoine	Fabian	Uncert. Data Mgmt	3
2017-12-12	Silviu	Fabian	Uncert. Data Mgmt	4

$\Pi_{\textbf{teacher},\textbf{resp}}(\text{Class})$

teacher	resp
Antoine	Fabian
Silviu	Fabian

→ Duplicates are removed

Selection: keep a subset of the tuples

Class

date	teacher	resp	name	num
2017-11-21	Antoine	Fabian	Uncert. Data Mgmt	1
2017-11-28	Antoine	Fabian	Uncert. Data Mgmt	2
2017-12-05	Antoine	Fabian	Uncert. Data Mgmt	3
2017-12-12	Silviu	Fabian	Uncert. Data Mgmt	4

Selection: keep a subset of the tuples

Class

date	teacher	resp	name	num
2017-11-21	Antoine	Fabian	Uncert. Data Mgmt	1
2017-11-28	Antoine	Fabian	Uncert. Data Mgmt	2
2017-12-05	Antoine	Fabian	Uncert. Data Mgmt	3
2017-12-12	Silviu	Fabian	Uncert. Data Mgmt	4

$\sigma_{\textbf{teacher}=\text{“Silviu”}}(\text{Class})$

date	teacher	resp	name	num

Selection: keep a subset of the tuples

Class

date	teacher	resp	name	num
2017-11-21	Antoine	Fabian	Uncert. Data Mgmt	1
2017-11-28	Antoine	Fabian	Uncert. Data Mgmt	2
2017-12-05	Antoine	Fabian	Uncert. Data Mgmt	3
2017-12-12	Silviu	Fabian	Uncert. Data Mgmt	4

$\sigma_{\textbf{teacher}=\text{“Silviu”}}(\text{Class})$

date	teacher	resp	name	num
2017-12-12	Silviu	Fabian	Uncert. Data Mgmt	4

Rename: change the name of attributes

Class

date	teacher	resp	name	num
2017-11-21	Antoine	Fabian	Uncert. Data Mgmt	1
2017-11-28	Antoine	Fabian	Uncert. Data Mgmt	2
2017-12-05	Antoine	Fabian	Uncert. Data Mgmt	3
2017-12-12	Silviu	Fabian	Uncert. Data Mgmt	4

Rename: change the name of attributes

Class

date	teacher	resp	name	num
2017-11-21	Antoine	Fabian	Uncert. Data Mgmt	1
2017-11-28	Antoine	Fabian	Uncert. Data Mgmt	2
2017-12-05	Antoine	Fabian	Uncert. Data Mgmt	3
2017-12-12	Silviu	Fabian	Uncert. Data Mgmt	4

$\rho_{\textbf{resp}\to\textbf{boss}}(\text{Class})$

Rename: change the name of attributes

Class

date	teacher	resp	name	num
2017-11-21	Antoine	Fabian	Uncert. Data Mgmt	1
2017-11-28	Antoine	Fabian	Uncert. Data Mgmt	2
2017-12-05	Antoine	Fabian	Uncert. Data Mgmt	3
2017-12-12	Silviu	Fabian	Uncert. Data Mgmt	4

$\rho_{\textbf{resp}\to\textbf{boss}}$(Class)

date	teacher	boss	name	num
2017-11-21	Antoine	Fabian	Uncert. Data Mgmt	1
2017-11-28	Antoine	Fabian	Uncert. Data Mgmt	2
2017-12-05	Antoine	Fabian	Uncert. Data Mgmt	3
2017-12-12	Silviu	Fabian	Uncert. Data Mgmt	4

Union

- Take tuples occurring in one of the input tables
- Applies to two tables with the same attributes

Union

- Take tuples occurring in one of the input tables
- Applies to two tables with the same attributes

Students1

id	name
1	Arthur Dent
2	Ford Prefect

Union

- Take tuples occurring in one of the input tables
- Applies to two tables with the same attributes

Students1

id	name
1	Arthur Dent
2	Ford Prefect

$\cup$

Union

- Take tuples occurring in one of the input tables
- Applies to two tables with the same attributes

Students1

id	name
1	Arthur Dent
2	Ford Prefect

$\cup$

S2

id	name
42	Zaphod B.

Union

- Take tuples occurring in one of the input tables
- Applies to two tables with the same attributes

Students1

id	name
1	Arthur Dent
2	Ford Prefect

$\cup$

S2

id	name
42	Zaphod B.

$=$

Union

- Take tuples occurring in one of the input tables
- Applies to two tables with the same attributes

Students1

id	name
1	Arthur Dent
2	Ford Prefect

$\cup$

S2

id	name
42	Zaphod B.

$=$

Students1 $\cup$ S2

id	name
1	Arthur Dent
2	Ford Prefect
42	Zaphod B.

Union

- Take tuples occurring in one of the input tables
- Applies to two tables with the same attributes

Students1

id	name
1	Arthur Dent
2	Ford Prefect

$\cup$

S2

id	name
42	Zaphod B.

$=$

Students1 $\cup$ S2

id	name
1	Arthur Dent
2	Ford Prefect
42	Zaphod B.

$\rightarrow$ Duplicates are removed here as well

Product

- Take all combinations of the input tables

Product

- Take all combinations of the input tables

Students

id	name
1	Arthur Dent
2	Ford Prefect

Product

- Take all combinations of the input tables

Students

id	name
1	Arthur Dent
2	Ford Prefect

Product

- Take all combinations of the input tables

Students

id	name
1	Arthur Dent
2	Ford Prefect

$\times$

Rooms

room
E200
E242

Product

- Take all combinations of the input tables

Students

id	name
1	Arthur Dent
2	Ford Prefect

$\times$

Rooms

room
E200
E242

$=$

Product

- Take all combinations of the input tables

Students

id	name
1	Arthur Dent
2	Ford Prefect

$\times$

Rooms

room
E200
E242

$=$

Students $\times$ Rooms

id	name	room
1	Arthur Dent	E200
1	Arthur Dent	E242
2	Ford Prefect	E200
2	Ford Prefect	E242

Join

→ Product is useful to express joins:

Join

→ Product is useful to express joins:

Member

id	class
1	UDM
2	UDM

Join

→ Product is useful to express joins:

Member

id	class
1	UDM
2	UDM

⋈

Join

→ Product is useful to express joins:

Member

id	class
1	UDM
2	UDM

$\bowtie$

Class

class	date
UDM	Nov 21
ABC	Nov 24
UDM	Nov 28

Join

→ Product is useful to express joins:

Member

id	class
1	UDM
2	UDM

$\bowtie$

Class

class	date
UDM	Nov 21
ABC	Nov 24
UDM	Nov 28

$=$

Join

→ Product is useful to express joins:

Member

id	class
1	UDM
2	UDM

⋈

Class

class	date
UDM	Nov 21
ABC	Nov 24
UDM	Nov 28

=

Member ⋈ Class

id	class	date
1	UDM	Nov 21
1	UDM	Nov 28
2	UDM	Nov 21
2	UDM	Nov 28

Join

→ Product is useful to express joins:

Member

id	class
1	UDM
2	UDM

$\bowtie$

Class

class	date
UDM	Nov 21
ABC	Nov 24
UDM	Nov 28

$=$

Member $\bowtie$ Class

id	class	date
1	UDM	Nov 21
1	UDM	Nov 28
2	UDM	Nov 21
2	UDM	Nov 28

Express Member $\bowtie$ Class with the previous operators:

Join

→ Product is useful to express joins:

Member

id	class
1	UDM
2	UDM

$\bowtie$

Class

class	date
UDM	Nov 21
ABC	Nov 24
UDM	Nov 28

$=$

Member $\bowtie$ Class

id	class	date
1	UDM	Nov 21
1	UDM	Nov 28
2	UDM	Nov 21
2	UDM	Nov 28

Express Member $\bowtie$ Class with the previous operators:

Member $\times$ Class

Join

→ Product is useful to express joins:

Member

id	class
1	UDM
2	UDM

$\bowtie$

Class

class	date
UDM	Nov 21
ABC	Nov 24
UDM	Nov 28

$=$

Member $\bowtie$ Class

id	class	date
1	UDM	Nov 21
1	UDM	Nov 28
2	UDM	Nov 21
2	UDM	Nov 28

Express Member $\bowtie$ Class with the previous operators:

$$\rho_{\textbf{class}\to\textbf{class2}}(\text{Member}) \times \text{Class}$$

Join

→ Product is useful to express joins:

Member

id	class
1	UDM
2	UDM

$\bowtie$

Class

class	date
UDM	Nov 21
ABC	Nov 24
UDM	Nov 28

$=$

Member $\bowtie$ Class

id	class	date
1	UDM	Nov 21
1	UDM	Nov 28
2	UDM	Nov 21
2	UDM	Nov 28

Express Member $\bowtie$ Class with the previous operators:

$$\sigma_{\textbf{class}=\textbf{class2}} \left(\rho_{\textbf{class}\to\textbf{class2}}(\text{Member}) \times \text{Class} \right)$$

Join

→ Product is useful to express joins:

Member

id	class
1	UDM
2	UDM

⋈

Class

class	date
UDM	Nov 21
ABC	Nov 24
UDM	Nov 28

=

Member ⋈ Class

id	class	date
1	UDM	Nov 21
1	UDM	Nov 28
2	UDM	Nov 21
2	UDM	Nov 28

Express Member ⋈ Class with the previous operators:

$$\Pi_{\text{id,class,date}}\Big(\ \sigma_{\textbf{class}=\textbf{class2}}\ \big(\ \rho_{\textbf{class}\to\textbf{class2}}(\text{Member}) \times \text{Class}\big)\Big)$$

Difference

- Take tuples that are in one table but not in the other
- Applies to two tables with same attributes

Difference

- Take tuples that are in one table but not in the other
- Applies to two tables with same attributes

Students1

id	name
1	Arthur Dent
2	Ford Prefect

Difference

- Take tuples that are in one table but not in the other
- Applies to two tables with same attributes

Students1

id	name
1	Arthur Dent
2	Ford Prefect

—

Difference

- Take tuples that are in one table but not in the other
- Applies to two tables with same attributes

Students1

id	name
1	Arthur Dent
2	Ford Prefect

—

S3

id	name
1	Arthur Dent
42	Zaphod B.

Difference

- Take tuples that are in one table but not in the other
- Applies to two tables with same attributes

Students1

id	name
1	Arthur Dent
2	Ford Prefect

$-$

S3

id	name
1	Arthur Dent
42	Zaphod B.

$=$

Difference

- Take tuples that are in one table but not in the other
- Applies to two tables with same attributes

Students1

id	name
1	Arthur Dent
2	Ford Prefect

$-$

S3

id	name
1	Arthur Dent
42	Zaphod B.

$=$

Students1 $-$ S3

id	name
2	Ford Prefect

Table of contents

Basic relations

```
CREATE TABLE Students(id INT(6), name VARCHAR(30));
INSERT INTO Students VALUES (1, 'Arthur Dent');
INSERT INTO Students VALUES (2, 'Ford Prefect');
```

Basic relations

```
CREATE TABLE Students(id INT(6), name VARCHAR(30));
INSERT INTO Students VALUES (1, 'Arthur Dent');
INSERT INTO Students VALUES (2, 'Ford Prefect');

SELECT * FROM Students;
```

Basic relations

```
CREATE TABLE Students(id INT(6), name VARCHAR(30));
INSERT INTO Students VALUES (1, 'Arthur Dent');
INSERT INTO Students VALUES (2, 'Ford Prefect');

SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
2 rows in set (0.00 sec)
```

Projection and rename

```
SELECT name FROM Students;
```

Projection and rename

```
SELECT name FROM Students;
```

```
+--------------+
| name         |
+--------------+
| Arthur Dent  |
| Ford Prefect |
+--------------+
```

Projection and rename

```
SELECT name FROM Students;
```

```
+--------------+
| name         |
+--------------+
| Arthur Dent  |
| Ford Prefect |
+--------------+
```

```
SELECT name, id, id AS identifier FROM Students;
```

Projection and rename

```
SELECT name FROM Students;
```

```
+--------------+
| name         |
+--------------+
| Arthur Dent  |
| Ford Prefect |
+--------------+
```

```
SELECT name, id, id AS identifier FROM Students;
```

```
+--------------+------+------------+
| name         | id   | identifier |
+--------------+------+------------+
| Arthur Dent  |    1 |          1 |
| Ford Prefect |    2 |          2 |
+--------------+------+------------+
```

Selection

```sql
SELECT * FROM Students;
```

Selection

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

Selection

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

```
SELECT * FROM Students WHERE id='2';
```

Selection

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

```
SELECT * FROM Students WHERE id='2';
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    2 | Ford Prefect |
+------+--------------+
```

Union

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

Union

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

```
SELECT * FROM S2;
```

```
+------+----------+
| id   | name     |
+------+----------+
|   42 | Zaphod B |
+------+----------+
```

Union

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

```
SELECT * FROM S2;
```

```
+------+----------+
| id   | name     |
+------+----------+
|   42 | Zaphod B |
+------+----------+
```

```
(SELECT * FROM Students)
UNION
(SELECT * FROM S2);
```

Union

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

```
SELECT * FROM S2;
```

```
+------+----------+
| id   | name     |
+------+----------+
|   42 | Zaphod B |
+------+----------+
```

```
(SELECT * FROM Students)
UNION
(SELECT * FROM S2);
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
|   42 | Zaphod B     |
+------+--------------+
```

Product

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

Product

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

```
SELECT * FROM Rooms;
```

```
+------+
| room |
+------+
| E200 |
| E242 |
+------+
```

Product

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

```
SELECT * FROM Rooms;
```

```
+------+
| room |
+------+
| E200 |
| E242 |
+------+
```

```
SELECT * FROM Students, Rooms;
```

Product

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

```
SELECT * FROM Rooms;
```

```
+------+
| room |
+------+
| E200 |
| E242 |
+------+
```

```
SELECT * FROM Students, Rooms;
```

```
+------+--------------+------+
| id   | name         | room |
+------+--------------+------+
|    1 | Arthur Dent  | E200 |
|    2 | Ford Prefect | E200 |
|    1 | Arthur Dent  | E242 |
|    2 | Ford Prefect | E242 |
+------+--------------+------+
```

Join

```
SELECT * FROM Member;
```

```
+------+-------+
| id   | class |
+------+-------+
|    1 | UDM   |
|    2 | UDM   |
+------+-------+
```

Join

```
SELECT * FROM Member;
```

```
+------+-------+
| id   | class |
+------+-------+
|    1 | UDM   |
|    2 | UDM   |
+------+-------+
```

```
SELECT * FROM Classes;
```

```
+-------+--------+
| class | date   |
+-------+--------+
| UDM   | Nov 21 |
| ABC   | Nov 24 |
| UDM   | Nov 28 |
+-------+--------+
```

Join

```
SELECT * FROM Member;
```

```
+------+-------+
| id   | class |
+------+-------+
|    1 | UDM   |
|    2 | UDM   |
+------+-------+
```

```
SELECT * FROM Classes;
```

```
+-------+--------+
| class | date   |
+-------+--------+
| UDM   | Nov 21 |
| ABC   | Nov 24 |
| UDM   | Nov 28 |
+-------+--------+
```

```
SELECT * FROM
  Member NATURAL JOIN Classes;
```

Join

```
SELECT * FROM Member;
```

```
+------+-------+
| id   | class |
+------+-------+
|    1 | UDM   |
|    2 | UDM   |
+------+-------+
```

```
SELECT * FROM Classes;
```

```
+-------+--------+
| class | date   |
+-------+--------+
| UDM   | Nov 21 |
| ABC   | Nov 24 |
| UDM   | Nov 28 |
+-------+--------+
```

```
SELECT * FROM
  Member NATURAL JOIN Classes;
```

```
+-------+------+--------+
| class | id   | date   |
+-------+------+--------+
| UDM   |    1 | Nov 21 |
| UDM   |    2 | Nov 21 |
| UDM   |    1 | Nov 28 |
| UDM   |    2 | Nov 28 |
+-------+------+--------+
```

Difference

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

Difference

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

```
SELECT * FROM S3;
```

```
+------+-------------+
| id   | name        |
+------+-------------+
|    1 | Arthur Dent |
|   42 | Zaphod B.   |
+------+-------------+
```

Difference

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

```
SELECT * FROM S3;
```

```
+------+-------------+
| id   | name        |
+------+-------------+
|    1 | Arthur Dent |
|   42 | Zaphod B.   |
+------+-------------+
```

```
SELECT * FROM Students
WHERE (id, name) NOT IN
  (SELECT * FROM S3);
```

Difference

```
SELECT * FROM Students;
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    1 | Arthur Dent  |
|    2 | Ford Prefect |
+------+--------------+
```

```
SELECT * FROM S3;
```

```
+------+------------+
| id   | name       |
+------+------------+
|    1 | Arthur Dent |
|   42 | Zaphod B.  |
+------+------------+
```

```
SELECT * FROM Students
WHERE (id, name) NOT IN
  (SELECT * FROM S3);
```

```
+------+--------------+
| id   | name         |
+------+--------------+
|    2 | Ford Prefect |
+------+--------------+
```

Composing operations

Our translation of:

```
SELECT * FROM
  Member NATURAL JOIN Classes;
```

can be expressed as:

Composing operations

Our translation of:

```
SELECT * FROM
  Member NATURAL JOIN Classes;
```

can be expressed as:

```
SELECT id, class, date FROM
  (SELECT * FROM
    (SELECT id, class AS class2 FROM Member) sub1,
    Classes
  ) sub2
  WHERE class = class2;
```

Composing operations

Our translation of:

```
SELECT * FROM
  Member NATURAL JOIN Classes;
```

can be expressed as:

```
SELECT id, class, date FROM
  (SELECT * FROM
    (SELECT id, class AS class2 FROM Member) sub1,
    Classes
  ) sub2
  WHERE class = class2;
```

→ SQL can express the relational algebra

Composing operations

Our translation of:

```
SELECT * FROM
  Member NATURAL JOIN Classes;
```

can be expressed as:

```
SELECT id, class, date FROM
  (SELECT * FROM
    (SELECT id, class AS class2 FROM Member) sub1,
    Classes
  ) sub2
  WHERE class = class2;
```

→ SQL can express the relational algebra

- … but please, never write queries like this!

References i

Abiteboul, S., Hull, R., and Vianu, V. (1995).

Foundations of Databases.

Addison-Wesley.

`http://webdam.inria.fr/Alice/pdfs/all.pdf`.

ISO (2008).

ISO 9075:2008: SQL.