Simple Example

All records for students more than 18 years old

$$\{\, S \mid S \in \text{Students} \,\wedge\, S.\text{age} > 18 \,\}$$

The set of tuples S such that S is in the table "Students" and has component "age" least 18.

This is like list comprehension in Haskell

```
[ s | s <- students, age s > 18 ]
```

and similar constructions in other languages.

All are based on "comprehensions" in set theory

Tuple Relational Calculus Basics

Queries in TRC have the general form

$$\{ \mathrm{T} \mid \mathrm{P}(\mathrm{T}) \}$$

where T is a *tuple variable* and $\mathrm{P}(\mathrm{T})$ is a logical formula.

Every tuple variable such as T has a *schema*, like rows in a relational table, with fields and their domains. In practice, the details of the schema are usually inferred from the way T appears in $\mathrm{P}(\mathrm{T})$.

A tuple variable ranges over all possible tuple values matching its schema.

The result of the query

$$\{ \mathrm{T} \mid \mathrm{P}(\mathrm{T}) \}$$

is then the set of all possible tuple values for T such that $\mathrm{P}(\mathrm{T})$ is true.

Another Example

Names and ages of all students over 20

$$\{\, \mathrm{T} \mid \exists S \,.\, S \in \mathrm{Students} \;\wedge\; S.\mathrm{age} > 20 \;\wedge\; \mathrm{T.name} = S.\mathrm{name} \;\wedge\; \mathrm{T.age} = S.\mathrm{age} \,\}$$

The set of tuples T such that there is an S in table “Students” with component “age” at least 20 and where S and T have the same values for “name” and “age”.

- Tuple variable S has schema matching the table “Students”.
- Tuple variable T has (only) fields “name” and “age”, with domains to match those of S.
- Even if S has other fields, they do not appear in T or the overall result.

Formula Syntax

Inside TRC expression $\{T \mid P(T)\}$ the logical formula $P(T)$ may be quite long, but is built up from standard logical components.

- Simple assertions: $(T \in \text{Table})$, $(T.\text{age} > 65)$, $(S.\text{name} = T.\text{name})$, ...
- Logical combinations: $(P \vee Q)$, $(P \wedge Q \wedge \neg Q')$, ...
- Quantification:

$\exists S \,.\, P(S)$	There exists a tuple S such that $P(S)$
$\forall T \,.\, Q(T)$	For all tuples T it is true that $Q(T)$

For convenience, we require that for $\exists S \,.\, P(S)$ the variable S must actually appear in $P(S)$; and the same for $\forall T \,.\, Q(T)$. We also write:

$$\exists S \in \text{Table} \,.\, P(S) \quad \text{to mean} \quad \exists S \,.\, S \in \text{Table} \wedge P(S)$$

Students and Courses (1/5)

mn	code	mark
s0456782	inf1	80
s0412375	geo1	78
s0412375	inf1	56
s0189034	math1	62

Referencing relation

mn	name	age	email
s0456782	John	18	john@inf
s0412375	Mary	18	mary@inf
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

code	name	year
inf1	Informatics 1	1
math1	Mathematics 1	1
geo1	Geology 1	1
dbs	Database Systems	3
adbs	Advanced Databases	4

Referenced relations

Primary key

Foreign key

Students and Courses (1/5)

Students taking Informatics 1

$$\{\, R \mid \exists S \in \text{Students} \,.\, \exists T \in \text{Takes} \,.\, \exists C \in \text{Courses} \,.\, C.\text{name} = \text{"Informatics 1"} \;\wedge\; C.\text{code} = T.\text{code} \;\wedge\; T.\text{mn} = S.\text{mn} \;\wedge\; S.\text{name} = R.\text{name} \,\}$$

Schema for S, T and C match those of the tables from which they are drawn. The schema for result R is a single field "name" with string domain, because that's all that appears here.

One way to compute this in relational algebra:

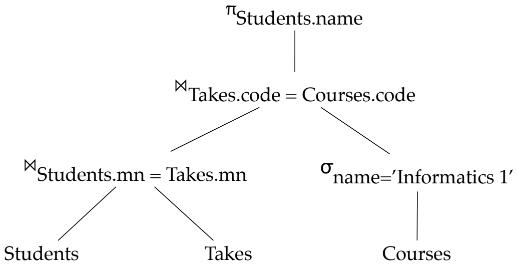
$$\pi_{\text{name}}((\text{Students} \bowtie \text{Takes}) \bowtie (\sigma_{\text{name}=\text{"Informatics 1"}}(\text{Courses})))$$

Relational Algebra

The relational algebra expression can be rearranged without changing its value, but possibly affecting the time and memory needed for computation:

$$\pi_{\mathsf{name}}((\mathsf{Students} \bowtie \mathsf{Takes}) \bowtie (\sigma_{\mathsf{name}=\texttt{"Informatics 1"}}(\mathsf{Courses})))$$
$$\pi_{\mathsf{name}}(\mathsf{Students} \bowtie (\mathsf{Takes} \bowtie (\sigma_{\mathsf{name}=\texttt{"Informatics 1"}}(\mathsf{Courses}))))$$
$$\pi_{\mathsf{name}}(\mathsf{Students} \bowtie ((\sigma_{\mathsf{name}=\texttt{"Informatics 1"}}(\mathsf{Courses})) \bowtie \mathsf{Takes}))$$

We can also visualise this as rearrangements of a tree:



Students and Courses (2/5)

Courses taken by students called "Joe"

$$\{\, R \mid \exists S \in \text{Students}, T \in \text{Takes}, C \in \text{Courses}\ .$$
$$S.\text{name} = \text{"Joe"} \wedge S.\text{mn} = T.\text{mn}$$
$$\wedge\ C.\text{code} = T.\text{code} \wedge C.\text{name} = R.\text{name} \,\}$$

Note the slightly abbreviated syntax for multiple quantification: we use comma-separated $\exists .., .., ..$ instead of $\exists .. \exists .. \exists ..$

Computing this in relational algebra:

$$\pi_{\text{name}}((\text{Courses} \bowtie \text{Takes}) \bowtie (\sigma_{\text{name}=\text{"Joe"}}(\text{Students})))$$

Students and Courses (3/5)

Students taking Informatics 1 or Geology 1

$$
\begin{aligned}
\{\, R \mid \exists S \in \text{Students}, T \in \text{Takes}, C \in \text{Courses}\ .\\
(C.\text{name} = \text{"Informatics 1"} \lor C.\text{name} = \text{"Geology 1"})\\
\land\ C.\text{code} = T.\text{code}\ \land\ T.\text{mn} = S.\text{mn}\ \land\ S.\text{name} = R.\text{name}\,\}
\end{aligned}
$$

Now the logical formula becomes a little more elaborate.

Computing this in relational algebra:

$$
\begin{aligned}
&\pi_{\text{name}}((\text{Students} \bowtie \text{Takes}) \bowtie (\sigma_{\text{name}=\text{"Informatics 1"}}(\text{Courses})))\\
&\qquad \cup \pi_{\text{name}}((\text{Students} \bowtie \text{Takes}) \bowtie (\sigma_{\text{name}=\text{"Geology 1"}}(\text{Courses})))
\end{aligned}
$$

$$
\pi_{\text{name}}((\text{Students} \bowtie \text{Takes}) \bowtie (\sigma_{(\text{name}=\text{"Informatics 1"} \lor \text{name}=\text{"Geology 1"})}(\text{Courses}))
$$

Students and Courses (4/5)

Students taking both Informatics 1 and Geology 1

$$
\begin{aligned}
\{\, R \mid \exists S \in \mathsf{Students}, T, T' \in \mathsf{Takes}, C, C' \in \mathsf{Courses}\ .\\
&C.\mathsf{name} = \texttt{"Informatics 1"} \ \wedge\ C.\mathsf{code} = T.\mathsf{code} \ \wedge\ T.\mathsf{mn} = S.\mathsf{mn}\\
&C'.\mathsf{name} = \texttt{"Geology 1"} \ \wedge\ C'.\mathsf{code} = T'.\mathsf{code} \ \wedge\ T'.\mathsf{mn} = S.\mathsf{mn}\\
&\ \wedge\ S.\mathsf{name} = R.\mathsf{name} \,\}
\end{aligned}
$$

Computing this in relational algebra:

$$
\begin{aligned}
&\pi_{\mathsf{name}}((\mathsf{Students} \bowtie \mathsf{Takes}) \bowtie (\sigma_{\mathsf{name}=\texttt{"Informatics 1"}}(\mathsf{Courses})))\\
&\qquad \cap\, \pi_{\mathsf{name}}((\mathsf{Students} \bowtie \mathsf{Takes}) \bowtie (\sigma_{\mathsf{name}=\texttt{"Geology 1"}}(\mathsf{Courses})))
\end{aligned}
$$

Students and Courses (5/5)

Students taking no courses

$$\{\, R \mid \exists S \in \mathsf{Students} \,.\, S.\mathsf{name} = R.\mathsf{name} \;\wedge\; \forall T \in \mathsf{Takes} \,.\, T.\mathsf{mn} \neq S.\mathsf{mn}$$

Computing this in relational algebra:

$$\pi_{\mathsf{name}}(\mathsf{Students} - \pi_{\mathsf{name,mn}}(\mathsf{Students} \bowtie \mathsf{Takes}))$$

⋆ Challenge: why not one of these instead?

$$\pi_{\mathsf{name}}(\mathsf{Students} - (\mathsf{Students} \bowtie \mathsf{Takes}))$$

$$\pi_{\mathsf{name}}(\mathsf{Students}) - \pi_{\mathsf{name}}(\mathsf{Students} \bowtie \mathsf{Takes}))$$