Databases and Information Systems

Relational Calculus
Domain Independence

Dmitriy Traytel

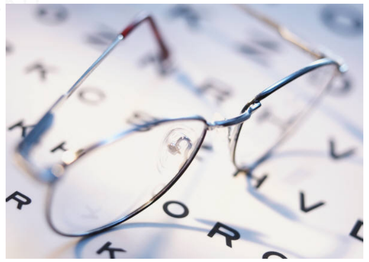
Do-It-Yourself Recap: Relational Calculus

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\exists ssn1, ssn2, lot1, lot2.$$
$$Employees(ssn1,name1,lot1) \wedge Employees(ssn2,name2,lot2)$$
$$\wedge\ lot1 \approx lot2 \wedge \neg\ ssn1 \approx ssn2$$

What should we learn today?



- Understand the semantics of the relational calculus
- Formulate queries in the relational calculus (this time for real)
- Understand the notion of domain independence and be able to argue whether a given query is domain independent
- Understand the relational algebra normal form (RANF) and be able to determine whether a relational calculus query is in RANF

Relational Calculus Semantics

Fix a database (mapping of table names to relations) DB

$$
\begin{aligned}
v \models P(t1, \ldots, tn) &\iff (v(t1), \ldots, v(tn)) \in DB(P) \\
v \models t1 \approx t2 &\iff v(t1) = v(t2) \\
v \models \neg\, \varphi &\iff v \not\models \varphi \\
v \models \varphi \vee \psi &\iff v \models \varphi \text{ or } v \models \psi \\
v \models \varphi \wedge \psi &\iff v \models \varphi \text{ and } v \models \psi \\
v \models \forall x.\ \varphi &\iff v(x \mapsto c) \models \varphi \text{ for all } c \in \mathbb{D} \\
v \models \exists x.\ \varphi &\iff v(x \mapsto c) \models \varphi \text{ for some } c \in \mathbb{D}
\end{aligned}
$$

Relational Calculus Semantics

Fix a database (mapping of table names to relations) DB

Maps variables to domain elements

$$
\begin{aligned}
v \models P(t1, \ldots, tn) &\iff (v(t1), \ldots, v(tn)) \in DB(P) \\
v \models t1 \approx t2 &\iff v(t1) = v(t2) \\
v \models \neg\, \varphi &\iff v \not\models \varphi \\
v \models \varphi \vee \psi &\iff v \models \varphi \text{ or } v \models \psi \\
v \models \varphi \wedge \psi &\iff v \models \varphi \text{ and } v \models \psi \\
v \models \forall x.\ \varphi &\iff v(x \mapsto c) \models \varphi \text{ for all } c \in \mathbb{D} \\
v \models \exists x.\ \varphi &\iff v(x \mapsto c) \models \varphi \text{ for some } c \in \mathbb{D}
\end{aligned}
$$

Predicates

$$v \models P(t1, \ldots, tn) \iff (v(t1), \ldots, v(tn)) \in DB(P)$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Predicates

$$v \vDash P(t1, \ldots, tn) \iff (v(t1), \ldots, v(tn)) \in DB(P)$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{ \begin{array}{l} ssn \mapsto 0983763423 \\ name \mapsto John \\ lot \mapsto 10 \end{array} \right\} \vDash Employees(ssn,name,lot)$$

Predicates

$$v \models P(t1, \ldots, tn) \iff (v(t1), \ldots, v(tn)) \in DB(P)$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{ \begin{array}{l} ssn \mapsto 0983763423 \\ name \mapsto John \\ lot \mapsto 10 \end{array} \right\} \models Employees(ssn,name,lot)$$

$$\left\{ \begin{array}{l} ssn \mapsto 3743923483 \\ name \mapsto Jill \end{array} \right\} \models Employees(ssn,name,20)$$

Predicates

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$v \models P(t1, \ldots, tn) \iff (v(t1), \ldots, v(tn)) \in DB(P)$$

$$\{ ssn \mapsto 0983763423,\ name \mapsto John,\ lot \mapsto 10 \} \models Employees(ssn,name,lot)$$

$$\{ ssn \mapsto 3743923483,\ name \mapsto Jill \} \models Employees(ssn,name,20)$$

$$\{ ssn \mapsto 0983763423,\ name \mapsto John \} \not\models Employees(ssn,name,20)$$

Equality

$$v \models t1 \approx t2 \iff v(t1) = v(t2)$$

Equality

$$v \models t1 \approx t2 \iff v(t1) = v(t2)$$

$$\{ lot \mapsto 10 \} \models lot \approx 10$$

Equality

$$v \models t1 \approx t2 \iff v(t1) = v(t2)$$

$$\{ lot \mapsto 10 \} \models lot \approx 10$$

$$\{ lot \mapsto 10 \} \not\models lot \approx 20$$

Equality

$$v \models t1 \approx t2 \iff v(t1) = v(t2)$$

$$\{ lot \mapsto 10 \} \models lot \approx 10$$

$$\{ lot \mapsto 10 \} \not\models lot \approx 20$$

$$\{ lot \mapsto 10, lot' \mapsto 20 \} \not\models lot \approx lot'$$

Negation

$v \models \neg \varphi \iff v \not\models \varphi$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Negation

$$\nu \models \neg\, \varphi \iff \nu \not\models \varphi$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{10} \end{array} \right\} \not\models \neg\, \texttt{Employees(ssn,name,lot)}$$

Negation

$$\nu \models \neg\, \varphi \iff \nu \not\models \varphi$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{10} \end{array} \right\} \not\models \neg\, \texttt{Employees(ssn,name,lot)}$$

$$\{\, \texttt{lot} \mapsto \texttt{10} \,\} \models \neg\, \texttt{lot} \approx \texttt{20}$$

Negation

$$\nu \models \neg\, \varphi \iff \nu \not\models \varphi$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{10} \end{array} \right\} \not\models \neg\, \texttt{Employees(ssn,name,lot)}$$

$$\{\, \texttt{lot} \mapsto \texttt{10} \,\} \models \neg\, \texttt{lot} \approx \texttt{20}$$

$$\{\, \texttt{lot} \mapsto \texttt{20} \,\} \not\models \neg\, \texttt{lot} \approx \texttt{20}$$

Disjunction

$$\nu \models \varphi \vee \psi \iff \nu \models \varphi \text{ or } \nu \models \psi$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Disjunction

$$\nu \models \varphi \vee \psi \iff \nu \models \varphi \text{ or } \nu \models \psi$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{10} \end{array} \right\} \models \texttt{Employees(ssn,name,lot)} \vee \texttt{lot} \approx \texttt{20}$$

Disjunction

$\nu \models \varphi \vee \psi \iff \nu \models \varphi$ or $\nu \models \psi$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{10} \end{array} \right\} \models \texttt{Employees(ssn,name,lot)} \vee \texttt{lot} \approx \texttt{20}$$

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{20} \end{array} \right\} \models \texttt{Employees(ssn,name,lot)} \vee \texttt{lot} \approx \texttt{20}$$

Disjunction

$\nu \models \varphi \vee \psi \iff \nu \models \varphi \text{ or } \nu \models \psi$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{\begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{10} \end{array}\right\} \models \texttt{Employees(ssn,name,lot)} \vee \texttt{lot} \approx \texttt{20}$$

$$\left\{\begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{20} \end{array}\right\} \models \texttt{Employees(ssn,name,lot)} \vee \texttt{lot} \approx \texttt{20}$$

$$\left\{\begin{array}{l} \texttt{ssn} \mapsto \texttt{0909090909} \\ \texttt{name} \mapsto \texttt{Dmitriy} \\ \texttt{lot} \mapsto \texttt{20} \end{array}\right\} \models \texttt{Employees(ssn,name,lot)} \vee \texttt{lot} \approx \texttt{20}$$

Implication

$$\varphi \rightarrow \psi := (\neg\varphi) \vee \psi$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Implication

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\varphi \to \psi := (\neg\varphi) \vee \psi$$

$$\left\{\begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{10} \end{array}\right\} \models \texttt{Employees(ssn,name,lot)} \to \texttt{lot} \approx \texttt{10}$$

Implication

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\varphi \to \psi := (\neg\varphi) \vee \psi$$

$$\left\{\begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{10} \end{array}\right\} \models \texttt{Employees(ssn,name,lot)} \to \texttt{lot} \approx \texttt{10}$$

$$\left\{\begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{20} \end{array}\right\} \models \texttt{Employees(ssn,name,lot)} \to \texttt{lot} \approx \texttt{10}$$

Implication

$$\varphi \to \psi := (\neg\varphi) \vee \psi$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{\begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{10} \end{array}\right\} \models \texttt{Employees(ssn,name,lot)} \to \texttt{lot} \approx \texttt{10}$$

$$\left\{\begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{20} \end{array}\right\} \models \texttt{Employees(ssn,name,lot)} \to \texttt{lot} \approx \texttt{10}$$

$$\left\{\begin{array}{l} \texttt{ssn} \mapsto \texttt{0909090909} \\ \texttt{name} \mapsto \texttt{Dmitriy} \\ \texttt{lot} \mapsto \texttt{20} \end{array}\right\} \not\models (\neg\ \texttt{Employees(ssn,name,lot)}) \to \texttt{lot} \approx \texttt{30}$$

Disjunction

$$\nu \models \varphi \vee \psi \iff \nu \models \varphi \text{ or } \nu \models \psi$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Disjunction

$$\nu \models \varphi \vee \psi \iff \nu \models \varphi \text{ or } \nu \models \psi$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{10} \end{array} \right\} \models \texttt{Employees(ssn,name,lot)} \vee \texttt{lot} \approx \texttt{20}$$

Disjunction

$$\nu \models \varphi \vee \psi \iff \nu \models \varphi \text{ or } \nu \models \psi$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{10} \end{array} \right\} \models \texttt{Employees(ssn,name,lot)} \vee \texttt{lot} \approx \texttt{20}$$

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{20} \end{array} \right\} \models \texttt{Employees(ssn,name,lot)} \vee \texttt{lot} \approx \texttt{20}$$

Disjunction

$$\nu \models \varphi \vee \psi \iff \nu \models \varphi \text{ or } \nu \models \psi$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{\begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{10} \end{array}\right\} \models \texttt{Employees(ssn,name,lot)} \vee \texttt{lot} \approx \texttt{20}$$

$$\left\{\begin{array}{l} \texttt{ssn} \mapsto \texttt{0983763423} \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto \texttt{20} \end{array}\right\} \models \texttt{Employees(ssn,name,lot)} \vee \texttt{lot} \approx \texttt{20}$$

$$\left\{\begin{array}{l} \texttt{ssn} \mapsto \texttt{0909090909} \\ \texttt{name} \mapsto \texttt{Dmitriy} \\ \texttt{lot} \mapsto \texttt{20} \end{array}\right\} \models \texttt{Employees(ssn,name,lot)} \vee \texttt{lot} \approx \texttt{20}$$

Existential Quantifier

$v \models \exists x.\ \varphi \iff v(x \mapsto c) \models \varphi$ for some $c \in \mathbb{D}$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Existential Quantifier

$\nu \models \exists x.\ \varphi \iff \nu(x \mapsto c) \models \varphi$ for some $c \in \mathbb{D}$

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto 0983763423 \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto 10 \end{array} \right\} \models \texttt{Employees(ssn,name,lot)}$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Existential Quantifier

$\nu \models \exists x.\ \varphi \iff \nu(x \mapsto c) \models \varphi$ for some $c \in \mathbb{D}$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto 0983763423 \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto 10 \end{array} \right\} \models \texttt{Employees(ssn,name,lot)}$$

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto 0983763423 \\ \texttt{name} \mapsto \texttt{John} \end{array} \right\} \models \exists \texttt{lot}.\ \texttt{Employees(ssn,name,lot)}$$

Existential Quantifier

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$v \models \exists x.\ \varphi \iff v(x \mapsto c) \models \varphi$ for some $c \in \mathbb{D}$

$$\{ \text{ssn} \mapsto 0983763423,\ \text{name} \mapsto \text{John},\ \text{lot} \mapsto 10 \} \models \text{Employees(ssn,name,lot)}$$

$$\{ \text{ssn} \mapsto 0983763423,\ \text{name} \mapsto \text{John} \} \models \exists \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{ \text{name} \mapsto \text{John} \} \models \exists \text{ssn}.\ \exists \text{lot}.\ \text{Employees(ssn,name,lot)}$$

Existential Quantifier

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$v \models \exists x.\ \varphi \iff v(x \mapsto c) \models \varphi$ for some $c \in \mathbb{D}$

$$\{ \text{ssn} \mapsto 0983763423,\ \text{name} \mapsto \text{John},\ \text{lot} \mapsto 10 \} \models \text{Employees(ssn,name,lot)}$$

$$\{ \text{ssn} \mapsto 0983763423,\ \text{name} \mapsto \text{John} \} \models \exists \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{ \text{name} \mapsto \text{John} \} \models \exists \text{ssn}.\ \exists \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{ \} \models \exists \text{name}.\ \exists \text{ssn}.\ \exists \text{lot}.\ \text{Employees(ssn,name,lot)}$$

Existential Quantifier

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$v \models \exists x.\ \varphi \iff v(x \mapsto c) \models \varphi$ for some $c \in \mathbb{D}$

$$\left\{ \begin{array}{l} \text{ssn} \mapsto 0983763423 \\ \text{name} \mapsto \text{John} \\ \text{lot} \mapsto 10 \end{array} \right\} \models \text{Employees(ssn,name,lot)}$$

$$\left\{ \begin{array}{l} \text{ssn} \mapsto 0983763423 \\ \text{name} \mapsto \text{John} \end{array} \right\} \models \exists \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{ \text{name} \mapsto \text{John} \} \models \exists \text{ssn}.\ \exists \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{ \} \models \exists \text{name}.\ \exists \text{ssn}.\ \exists \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{ \} \models \exists \text{lot}.\ \text{lot} \approx 20$$

Existential Quantifier

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$\nu \models \exists x.\ \varphi \iff \nu(x \mapsto c) \models \varphi$ for some $c \in \mathbb{D}$

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto 0983763423 \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto 10 \end{array} \right\} \models \texttt{Employees(ssn,name,lot)}$$

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto 0983763423 \\ \texttt{name} \mapsto \texttt{John} \end{array} \right\} \models \exists \texttt{lot}.\ \texttt{Employees(ssn,name,lot)}$$

$$\{ \texttt{name} \mapsto \texttt{John} \} \models \exists \texttt{ssn}.\ \exists \texttt{lot}.\ \texttt{Employees(ssn,name,lot)}$$

$$\{ \} \models \exists \texttt{name}.\ \exists \texttt{ssn}.\ \exists \texttt{lot}.\ \texttt{Employees(ssn,name,lot)}$$

$$\{ \} \models \exists \texttt{lot}.\ \texttt{lot} \approx 20$$

$$\{ \} \not\models \exists \texttt{lot}.\ \neg\ \texttt{lot} \approx \texttt{lot}$$

Universal Quantifier

$v \models \forall x.\ \varphi \iff v(x \mapsto c) \models \varphi$ for all $c \in \mathbb{D}$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Universal Quantifier

$\nu \models \forall x.\ \varphi \iff \nu(x \mapsto c) \models \varphi \text{ for all } c \in \mathbb{D}$

$$\left\{ \begin{array}{l} \texttt{ssn} \mapsto 0983763423 \\ \texttt{name} \mapsto \texttt{John} \\ \texttt{lot} \mapsto 20 \end{array} \right\} \not\models \texttt{Employees(ssn,name,lot)}$$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Universal Quantifier

$v \models \forall x.\ \varphi \iff v(x \mapsto c) \models \varphi$ for all $c \in \mathbb{D}$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{ \begin{array}{l} \text{ssn} \mapsto 0983763423 \\ \text{name} \mapsto \text{John} \\ \text{lot} \mapsto 20 \end{array} \right\} \not\models \text{Employees(ssn,name,lot)}$$

$$\left\{ \begin{array}{l} \text{ssn} \mapsto 0983763423 \\ \text{name} \mapsto \text{John} \end{array} \right\} \not\models \forall \text{lot}.\ \text{Employees(ssn,name,lot)}$$

Universal Quantifier

$\nu \models \forall x.\ \varphi \iff \nu(x \mapsto c) \models \varphi$ for all $c \in \mathbb{D}$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\left\{ \begin{array}{l} \text{ssn} \mapsto 0983763423 \\ \text{name} \mapsto \text{John} \\ \text{lot} \mapsto 20 \end{array} \right\} \not\models \text{Employees(ssn,name,lot)}$$

$$\left\{ \begin{array}{l} \text{ssn} \mapsto 0983763423 \\ \text{name} \mapsto \text{John} \end{array} \right\} \not\models \forall \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{ \text{name} \mapsto \text{John} \} \not\models \forall \text{ssn}.\ \forall \text{lot}.\ \text{Employees(ssn,name,lot)}$$

Universal Quantifier

$v \models \forall x.\ \varphi \iff v(x \mapsto c) \models \varphi$ for all $c \in \mathbb{D}$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$$\{ \text{ssn} \mapsto 0983763423,\ \text{name} \mapsto \text{John},\ \text{lot} \mapsto 20 \} \not\models \text{Employees(ssn,name,lot)}$$

$$\{ \text{ssn} \mapsto 0983763423,\ \text{name} \mapsto \text{John} \} \not\models \forall \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{ \text{name} \mapsto \text{John} \} \not\models \forall \text{ssn}.\ \forall \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{ \} \not\models \forall \text{name}.\ \forall \text{ssn}.\ \forall \text{lot}.\ \text{Employees(ssn,name,lot)}$$

Universal Quantifier

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$v \models \forall x.\ \varphi \iff v(x \mapsto c) \models \varphi$ for all $c \in \mathbb{D}$

$$\{ \text{ssn} \mapsto 0983763423,\ \text{name} \mapsto \text{John},\ \text{lot} \mapsto 20 \} \not\models \text{Employees(ssn,name,lot)}$$

$$\{ \text{ssn} \mapsto 0983763423,\ \text{name} \mapsto \text{John} \} \not\models \forall \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{ \text{name} \mapsto \text{John} \} \not\models \forall \text{ssn}.\ \forall \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{\ \} \not\models \forall \text{name}.\ \forall \text{ssn}.\ \forall \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{\ \} \not\models \forall \text{lot}.\ \text{lot} \approx 20$$

Universal Quantifier

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

$v \models \forall x.\ \varphi \iff v(x \mapsto c) \models \varphi$ for all $c \in \mathbb{D}$

$$\{ \text{ssn} \mapsto 0983763423,\ \text{name} \mapsto \text{John},\ \text{lot} \mapsto 20 \} \not\models \text{Employees(ssn,name,lot)}$$

$$\{ \text{ssn} \mapsto 0983763423,\ \text{name} \mapsto \text{John} \} \not\models \forall \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{ \text{name} \mapsto \text{John} \} \not\models \forall \text{ssn}.\ \forall \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{ \} \not\models \forall \text{name}.\ \forall \text{ssn}.\ \forall \text{lot}.\ \text{Employees(ssn,name,lot)}$$

$$\{ \} \not\models \forall \text{lot}.\ \text{lot} \approx 20$$

$$\{ \} \models \forall \text{lot}.\ \text{lot} \approx \text{lot}$$

Free Variables

$$
\begin{aligned}
\mathrm{fv}(P(t_1, \ldots, t_n)) &\iff \mathrm{fv}(t_1) \cup \ldots \cup \mathrm{fv}(t_n) \\
\mathrm{fv}(t_1 \approx t_2) &\iff \mathrm{fv}(t_1) \cup \mathrm{fv}(t_2) \\
\mathrm{fv}(\neg\, \varphi) &\iff \mathrm{fv}(\varphi) \\
\mathrm{fv}(\varphi \vee \psi) &\iff \mathrm{fv}(\varphi) \cup \mathrm{fv}(\psi) \\
\mathrm{fv}(\varphi \wedge \psi) &\iff \mathrm{fv}(\varphi) \cup \mathrm{fv}(\psi) \\
\mathrm{fv}(\forall x.\ \varphi) &\iff \mathrm{fv}(\varphi) - \{x\} \\
\mathrm{fv}(\exists x.\ \varphi) &\iff \mathrm{fv}(\varphi) - \{x\}
\end{aligned}
$$

We also write $\mathrm{fv}(\varphi)$ for the **ordered tuple** of the free variables of a formula

Query Result

$[\![\varphi]\!] := \{(v(x1),\ldots,v(xn)) \mid v \models \varphi \text{ and } fv(\varphi)=(x1,\ldots,xn)\}$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Query Result

$[\![\varphi]\!] := \{(v(x1),\ldots,v(xn)) \mid v \models \varphi \text{ and } fv(\varphi)=(x1,\ldots,xn)\}$

$[\![Employees(ssn,name,10)]\!] =$
$\{(0983763423, John), (9384392483,Jane)\}$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Query Result

$[\![\varphi]\!] := \{(v(x1),\ldots,v(xn)) \mid v \models \varphi \text{ and } fv(\varphi)=(x1,\ldots,xn)\}$

$[\![Employees(ssn,name,10)]\!] =$
$\{(0983763423, John), (9384392483,Jane)\}$

$[\![EXISTS\ ssn.\ Employees(ssn,name,10)]\!] =$
$\{(John), (Jane)\}$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Query Result

$[\![\varphi]\!] := \{(v(x1),\ldots,v(xn)) \mid v \models \varphi \text{ and } fv(\varphi)=(x1,\ldots,xn)\}$

$[\![Employees(ssn,name,10)]\!] =$
$\{(0983763423, John), (9384392483,Jane)\}$

$[\![EXISTS\ ssn.\ Employees(ssn,name,10)]\!] =$
$\{(John), (Jane)\}$

$[\![EXISTS\ ssn, name.\ Employees(ssn,name,10)]\!] = \{()\}$

Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Query Result

$[\![\varphi]\!] := \{(v(x1),\ldots,v(xn)) \mid v \models \varphi \text{ and } fv(\varphi)=(x1,\ldots,xn)\}$

$[\![Employees(ssn,name,10)]\!] =$
$\{(0983763423, John), (9384392483,Jane)\}$

$[\![EXISTS\ ssn.\ Employees(ssn,name,10)]\!] =$
$\{(John), (Jane)\}$

$[\![EXISTS\ ssn,\ name.\ Employees(ssn,name,10)]\!] = \{()\}$

$[\![EXISTS\ ssn,\ name.\ Employees(ssn,name,42)]\!] = \{\}$

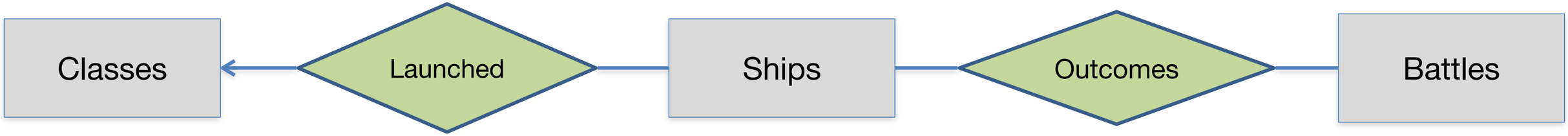
Employees

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Questions so far?

Case Study

E/R Battleships




```
Classes(class:string, type:string, country:string, numguns:int, bore:int, displacement:int)
Ships(name:string, class:string, launched:int)
Battles(name:string, date:string)
Outcomes(ship:string, battle:string, result:string)
```

Sample data

Sample data

Ships

name	class	launched
Alabama	South Dakota	1942
Missouri	Iowa	1944
Musashi	Yamato	1942
New Jersey	Iowa	1943
New Mexico	Mississippi	1918

Classes

class	type	country	num guns	bore	displacement
Bismarck	bb	Germany	8	15	42000
Iowa	bb	USA	9	16	46000
Kongo	bc	Japan	8	14	32000
South Dakota	bb	USA	9	16	37000
Renown	bc	Gt.Britain	6	15	32000

Battles

name	date
Denmark Strait	24-27.05.41
Guadalcanal	15.11.42
North Cape	26.12.43
Surigao Strait	25.10.44

Outcomes

ship	battle	result
Arizona	Pearl Habor	sunk
Bismarck	Denmark Strait	sunk
California	Suriago Strait	ok
Duke of York	North Cape	ok
Fuso	Suriago Strait	sunk

Query a

Give the class names and countries of the classes that carried guns of a 16-inch bore.

```
Classes(class:string, type:string, country:string, numguns:int, bore:int, displacement:int)
Ships(name:string, class:string, launched:int)
Battles(name:string, date:string)
Outcomes(ship:string, battle:string, result:string)
```

Query a

Give the class names and countries of the classes that carried guns of a 16-inch bore.

```
EXISTS t, n, b, d. Classes(cl,t,co,n,b,d) AND b = 16
```

```
Classes(class:string, type:string, country:string, numguns:int, bore:int, displacement:int)
Ships(name:string, class:string, launched:int)
Battles(name:string, date:string)
Outcomes(ship:string, battle:string, result:string)
```

Query b

Find the ships launched in 1921

```
Classes(class:string, type:string, country:string, numguns:int, bore:int, displacement:int)
Ships(name:string, class:string, launched:int)
Battles(name:string, date:string)
Outcomes(ship:string, battle:string, result:string)
```

Query b

Find the ships launched in 1921

```
EXISTS cl, l. Ships(n,cl,l) AND l = 1921
```

```
Classes(class:string, type:string, country:string, numguns:int, bore:int, displacement:int)
Ships(name:string, class:string, launched:int)
Battles(name:string, date:string)
Outcomes(ship:string, battle:string, result:string)
```

Query c

Find the ships sunk in the battle of the Denmark Strait

```
Classes(class:string, type:string, country:string, numguns:int, bore:int, displacement:int)
Ships(name:string, class:string, launched:int)
Battles(name:string, date:string)
Outcomes(ship:string, battle:string, result:string)
```

Query c

Find the ships sunk in the battle of the Denmark Strait

```
Outcomes(n,"DenmarkStrait","sunk")
```

```
Classes(class:string, type:string, country:string, numguns:int, bore:int, displacement:int)
Ships(name:string, class:string, launched:int)
Battles(name:string, date:string)
Outcomes(ship:string, battle:string, result:string)
```

Query d

List the name, displacement, and number of guns of the ships engaged in the battle of Guadalcanal.

```
Classes(class:string, type:string, country:string, numguns:int, bore:int, displacement:int)
Ships(name:string, class:string, launched:int)
Battles(name:string, date:string)
Outcomes(ship:string, battle:string, result:string)
```

Query d

List the name, displacement, and number of guns of the ships engaged in the battle of Guadalcanal.

```
EXISTS cl, l, t, co, b, r.
Ships(n,cl,l) AND Classes(cl,t,co,ng,b,d) AND
Outcomes(n,"Guadalcanal",r)
```

```
Classes(class:string, type:string, country:string, numguns:int, bore:int, displacement:int)
Ships(name:string, class:string, launched:int)
Battles(name:string, date:string)
Outcomes(ship:string, battle:string, result:string)
```

Query e

List all the ships mentioned in the database. (Recall that not all these ships appear in the Ships relation.)

```
Classes(class:string, type:string, country:string, numguns:int, bore:int, displacement:int)
Ships(name:string, class:string, launched:int)
Battles(name:string, date:string)
Outcomes(ship:string, battle:string, result:string)
```

Query e

List all the ships mentioned in the database. (Recall that not all these ships appear in the Ships relation.)

```
(EXISTS cl, l. Ships(n,cl,l)) OR
(EXISTS b, r. Outcomes(n,b,r))
```

```
Classes(class:string, type:string, country:string, numguns:int, bore:int, displacement:int)
Ships(name:string, class:string, launched:int)
Battles(name:string, date:string)
Outcomes(ship:string, battle:string, result:string)
```

Questions so far?

Revisiting Query e

List all the ships mentioned in the database. (Recall that not all these ships appear in the Ships relation.)

```
(EXISTS cl, l. Ships(n,cl,l)) OR
(EXISTS b, r. Outcomes(n,b,r))
```

```
Classes(class:string, type:string, country:string, numguns:int, bore:int, displacement:int)
Ships(name:string, class:string, launched:int)
Battles(name:string, date:string)
Outcomes(ship:string, battle:string, result:string)
```

Revisiting Query e

List all the ships mentioned in the database. (Recall that not all these ships appear in the Ships relation.)

```
EXISTS cl, l, b, r.
Ships(n,cl,l) OR Outcomes(n,b,r)
```

```
Classes(class:string, type:string, country:string, numguns:int, bore:int, displacement:int)
Ships(name:string, class:string, launched:int)
Battles(name:string, date:string)
Outcomes(ship:string, battle:string, result:string)
```

Revisiting Query e

List all the ships mentioned in the database. (Recall that not all these ships appear in the Ships relation.)

```
EXISTS cl, l, b, r.
Ships(n,cl,l) OR Outcomes(n,b,r)
```

```
Classes(class:string, type:string, country:string, numguns:int, bore:int, displacement:int)
Ships(name:string, class:string, launched:int)
Battles(name:string, date:string)
Outcomes(ship:string, battle:string, result:string)
```

Finite vs Infinite

- Fundamental problem with relational calculus:
  $[\![\phi]\!]$ is not always a finite relation

- Some examples for such "**unsafe**" queries
  - ϕ = Ships(n,cl,l) $\vee$ Outcomes(n,b,r)
  - ϕ = P(x) $\vee$ Q(y)
  - ϕ = $\neg$ P(x)
  - ϕ = x $\approx$ y
- But: query evaluation works with finite tables (why?)

Domain Independence

- All tables (predicates) are finite
- Q: Where does the infiniteness of $[\![\phi]\!]$ come from?

Domain Independence

- All tables (predicates) are finite
- Q: Where does the infiniteness of $[\![\phi]\!]$ come from?
- A: The domain $\mathbb{D}$

Domain Independence

- All tables (predicates) are finite
- Q: Where does the infiniteness of $[\![\phi]\!]$ come from?
- A: The domain $\mathbb{D}$
- $\mathbb{D}$ can be seen as a parameter of query evaluation: $[\![\phi]\!]_{\mathbb{D}}$

Domain Independence

- All tables (predicates) are finite
- Q: Where does the infiniteness of $[\![\phi]\!]$ come from?
- A: The domain $\mathbb{D}$
- $\mathbb{D}$ can be seen as a parameter of query evaluation: $[\![\phi]\!]_{\mathbb{D}}$
- ϕ is domain-independent if for all $\mathbb{D}$, $\mathbb{E}$: $[\![\phi]\!]_{\mathbb{D}} = [\![\phi]\!]_{\mathbb{E}}$

Domain Independence

- All tables (predicates) are finite
- Q: Where does the infiniteness of $[\![\phi]\!]$ come from?
- A: The domain $\mathbb{D}$
- $\mathbb{D}$ can be seen as a parameter of query evaluation: $[\![\phi]\!]_{\mathbb{D}}$
- ϕ is domain-independent if for all $\mathbb{D}$, $\mathbb{E}$: $[\![\phi]\!]_{\mathbb{D}} = [\![\phi]\!]_{\mathbb{E}}$
- For example: $P(x) \wedge Q(y)$ is domain-independent: $[\![P(x) \wedge Q(y)]\!]_{\mathbb{D}} = DB(P) \times DB(Q)$

Unsafe $\Longrightarrow$ Not Domain Independent

	$[\![\phi]\!]_{\mathbb{D}}$
$\phi = P(x) \vee Q(y)$	$\{(x,y).\ x \in DB(P)$ and $y \in \mathbb{D}$ or $x \in \mathbb{D}$ and $y \in DB(Q)\}$
$\phi = \neg P(x)$	$\{(x).\ x \in \mathbb{D}$ and $x \notin DB(P)\}$
$\phi = x \approx y$	$\{(x,x).\ x \in \mathbb{D}\}$

Safe $\Longrightarrow$ Domain Independent?

- Here, safe means “finite query result”
- Implication does **not** hold!

Safe $\Longrightarrow$ Domain Independent?

- Here, safe means “finite query result”
- Implication does **not** hold!
- Any formula without free variables (= closed) has a finite query result

Safe $\Longrightarrow$ Domain Independent?

- Here, safe means “finite query result”
- Implication does **not** hold!
- Any formula without free variables (= closed) has a finite query result
- $[\![\forall x.\ P(x)]\!]_{\mathbb{D}} = \{()\}$ if $\mathbb{D} = DB(P)$

Safe $\Longrightarrow$ Domain Independent?

- Here, safe means “finite query result”
- Implication does **not** hold!
- Any formula without free variables (= closed) has a finite query result
- $[\![\forall x.\ P(x)]\!]_{\mathbb{D}} = \{()\}$ if $\mathbb{D} = DB(P)$
- $[\![\forall x.\ P(x)]\!]_{\mathbb{D}} = \{\}$ if $DB(P) \subsetneq \mathbb{D}$

When is a formula domain-independent?

- Undecidable problem
  (= there exists no algorithm that answers the above question precisely)

- Resort to **syntactic** overapproximations:
  - under easy-to-check conditions a formula is domain-independent
    - e.g., the formula is a conjunction of n predicates (conjunctive queries)
  - conditions not met $\Longrightarrow$ the formula may or may not be domain-independent

Relational Algebra Normal Form

- A particular syntactic overapproximation
- RANF $\Longrightarrow$ domain-independent
- Even better: RANF $\Longrightarrow$ each “subformula” evaluates to a finite relation
- Has something to do with Relational Algebra (coming soon)

RANF (continued)

```
ranf(P(t1, …, tn))   ⟺ true
ranf(t1 ≈ t2)        ⟺ false
ranf(¬ φ)            ⟺ fv(φ)={} and ranf(φ)
ranf(φ ∨ ψ)          ⟺ ranf(φ) and ranf(ψ) and fv(φ)=fv(ψ)
ranf(φ ∧ ψ)          ⟺ …
ranf(∀x. φ)          ⟺ false
ranf(∃x. φ)          ⟺ ranf(φ)
```

RANF (continued)

ranf(φ) and ranf(ψ) or

ran

ran

ran

ranf(φ ∨ ψ) ⟺ anf(φ) and ranf(ψ) and fv(φ)=fv(ψ)

ranf(φ ∧ ψ) ⟺ …

ranf(∀x. φ) ⟺ false

ranf(∃x. φ) ⟺ ranf(φ)

RANF (continued)

ranf(φ) and ranf(ψ) or
ranf(φ) and ψ=¬χ and ranf(χ) and fv(χ)⊆fv(φ) or

ran
ran
ran

ranf(φ ∨ ψ) ⟺ anf(φ) and ranf(ψ) and fv(φ)=fv(ψ)
ranf(φ ∧ ψ) ⟺ …
ranf(∀x. φ) ⟺ false
ranf(∃x. φ) ⟺ ranf(φ)

RANF (continued)

ranf(φ) and ranf(ψ) or
ranf(φ) and ψ=¬χ and ranf(χ) and fv(χ)⊆fv(φ) or
ranf(φ) and ψ=t1 ≈ t2 and (fv(t1)⊆fv(φ) or fv(t2)⊆fv(φ)) or

ran
ran
ran
ranf(φ ∨ ψ) ⟺ anf(φ) and ranf(ψ) and fv(φ)=fv(ψ)
ranf(φ ∧ ψ) ⟺ …
ranf(∀x. φ) ⟺ false
ranf(∃x. φ) ⟺ ranf(φ)

RANF (continued)

```
ranf(φ) and ranf(ψ) or
ranf(φ) and ψ=¬χ and ranf(χ) and fv(χ)⊆fv(φ) or
ranf(φ) and ψ=t1 ≈ t2 and (fv(t1)⊆fv(φ) or fv(t2)⊆fv(φ)) or
ranf(φ) and ψ=¬t1 ≈ t2 and fv(t1)⊆fv(φ) and fv(t2)⊆fv(φ)
```

ran
ran
ran
ranf(φ ∨ ψ) ⟺ anf(φ) and ranf(ψ) and fv(φ)=fv(ψ)
ranf(φ ∧ ψ) ⟺ …
ranf(∀x. φ) ⟺ false
ranf(∃x. φ) ⟺ ranf(φ)

Codd's Theorem

For every domain-independent relational calculus query there exists an equivalent query in RANF

What should we learn today?

- Understand the semantics of the relational calculus
- Formulate queries in the relational calculus (this time for real)
- Understand the notion of domain independence and be able to argue whether a given query is domain independent
- Understand the relational algebra normal form (RANF) and be able to determine whether a relational calculus query is in RANF