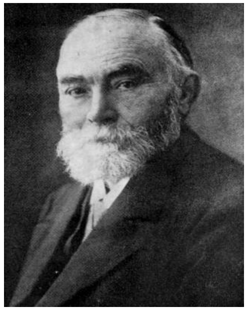
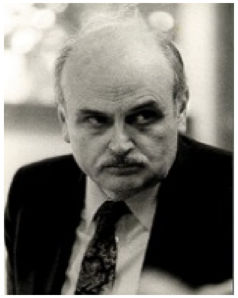
# Relational db: the origins



Frege: FO logic

Tarski: algebra for FO



Codd: relational databases

# Relational Calculus (aka FO)

- Models data manipulation core of SQL
  - idea: specify "what" not "how"

- General form defines the set of tuples t in the answer:

$$\{t \mid \text{property}\ (t)\}$$

- property (t) is described by a language based on predicate calculus (first-order logic)

# Reminder (CSE 20): some predicate calculus examples on natural numbers

- The set of even numbers:

$$\{ x \mid \exists y \, ( x = 2 * y)\}$$

- ? The set of prime numbers

$$\{x \mid x \neq 1 \wedge \forall y \, \forall z \, [ x = y * z \longrightarrow ((y = 1) \vee (z = 1))] \}$$

$\exists$ : "there exists" existential quantification
$\forall$ : "for all" universal quantification

# Relational calculus speaks about tuples

*Display the movie table*

SELECT *
FROM movie

In words (making answer tuple explicit):

"The answer consists of tuples m such that
m is a tuple in movie

Need to say:

"tuple m is in relation R": $m \in R$

# Examples

*Find the directors and actors of currently playing movies*

SELECT  m.Director, m.Actor
FROM  movie m, schedule s
WHERE  m.Title = s.Title

In words (making answer tuple explicit):

"The answer consists of tuples t such that
there exist tuples m in movie and s in schedule for which
t.Director = m.Director and t.Actor = m.Actor and
m.Title = s.Title"

Need to say:

"there exists a tuple x in relation R": $\exists\, x \in R$
Refer to the value of attribute A of tuple x: $x(A)$
Boolean combinations

# Examples (cont'd)

*Find the directors and actors of currently playing movies*

"The answer consists of tuples t such that
there exist tuples m in movie and s in schedule for which
t.Director = m.Director and t.Actor = m.Actor and
m.Title = s.Title"

In logic notation (tuple relational calculus):

$\{$ t: Director, Actor | $\exists$ m $\in$ movie $\exists$ s $\in$ schedule
[ t(Director) = m(Director) $\wedge$ t(Actor) = m(Actor)
$\wedge$ m(Title) = s(Title) ] $\}$

$\exists\ m \in R$ : existential quantification

"there exists some tuple m in relation R …."

Sometimes need to say

"for every tuple m …."

Example: "every director is also an actor"

Need to say:

"for every tuple m in movie there exists a tuple t in movie such that m.Director = t.Actor"

Logic notation: universal quantification $\forall\ m \in R$

$\forall\ m \in \text{movie}\ \exists\ t \in \text{movie}\ [\ m(\text{Director}) = t(\text{Actor})\ ]$

(The answer to this query is true or false)

# Tuple Relational Calculus

- In the style of SQL: language talks about tuples
- What you can say:
  - refer to tuples: tuple variables t, s, …
  - a tuple t belongs to a relation R: $t \in R$
  - conditions on attributes of a tuple t and s:
    - $t(A) = (\neq)(\geq)$ constant
    - $t(A) = s(B)$
    - $t(A) \neq s(B)$
    - etc.
- Simple expressions above: atoms

# Tuple Relational Calculus (2)

- Combine properties using Boolean operators
  - $\wedge, \vee, \neg$
  - (abbreviation: $p \rightarrow q \equiv \neg p \vee q$)
- Quantifiers
  - there exists: $\exists t \in R\ \varphi(t)$
  - for every: $\forall t \in R\ \varphi(t)$

  similar to local variable declarations

# More on quantifiers

- scope of quantifier:
  - scope of $\exists t \in R\ \varphi(t)$ is $\varphi$
  - scope of $\forall t \in R\ \varphi(t)$ is $\varphi$
- free variable:
  - not in scope of any quantifier
  - free variables are the “parameters” of the formula

# Examples

{ t: Director, Actor | $\exists$ m $\in$ movie $\exists$ s $\in$ schedule

[ t(Director) = m(Director) $\wedge$ t(Actor) = m(Actor) $\wedge$ m(Title) = s(Title) ] }

[ t(Director) = m(Director) $\wedge$ t(Actor) = m(Actor) $\wedge$ m(Title) = s(Title) ]

free: t, m, s

$\exists$ s $\in$ schedule

[ t(Director) = m(Director) $\wedge$ t(Actor) = m(Actor) $\wedge$ m(Title) = s(Title) ]

free: t, m

$\exists$ m $\in$ movie $\exists$ s $\in$ schedule

[ t(Director) = m(Director) $\wedge$ t(Actor) = m(Actor) $\wedge$ m(Title) = s(Title) ]

free: t

# Tuple Calculus Query

- {t: <att> | φ(t)}
  - where φ is a calculus formula
    with only one free variable t
  - produces as answer a table with attributes <att> consisting of all tuples v which make φ(v) true
  - Note: φ(v) has no free variables so it has no parameters and it evaluates to true or false
  - Range of answer tuple: usually specified in the query

  Otherwise, it is by default the active domain:

  set of values in database, or mentioned in query

# Examples (Movie Database)

- Find the titles of currently playing movies
  - $\{t: title \mid \exists s \in schedule\ [s(title) = t(title)]\}$
- Find the titles of movies by Berto
  - $\{t: title \mid \exists m \in movie\ [m(director) = \text{“Berto”} \wedge t(title) = m(title)]\}$
- Find the title and director of currently playing movies
  - $\{t: title, director \mid \exists s \in schedule\ \exists m \in movie\ [s(title) = m(title) \wedge t(title) = m(title) \wedge t(director) = m(director)]\}$

# Examples (max salary)

- Find employees with the highest salary:

| employee | name | salary |
|---|---|---|

$$\{x\text{: name} \mid \exists y \in \text{employee } [x(\text{name}) = y(\text{name}) \wedge \forall z \in \text{employee } (y(\text{salary}) \geq z(\text{salary}))]\}$$

# Examples (Movie Database)

- Find actors playing in every movie by Berto

$$\{a\text{: actor} \mid \exists y \in \text{movie}\,[a(\text{actor}) = y(\text{actor}) \wedge$$
$$\forall m \in \text{movie}\,[m(\text{director}) = \text{“Berto”} \rightarrow \exists t \in \text{movie}\,(m(\text{title}) =$$
$$t(\text{title}) \wedge t(\text{actor}) = y(\text{actor}))]]\}$$

# Examples (Movie Database)

- Find actors playing in every movie by Berto

$\{a\text{: actor} \mid \exists y \in \text{movie}\ [a(\text{actor}) = y(\text{actor}) \ \wedge$
$\forall m \in \text{movie}\ [m(\text{director}) = \text{“Berto”} \rightarrow \exists t \in \text{movie}\ (m(\text{title}) =$
$t(\text{title}) \wedge t(\text{actor}) = y(\text{actor}))]]\}$

Is the following correct?

$\{a\text{: actor} \mid \exists y \in \text{movie}\ [a(\text{actor}) = y(\text{actor}) \ \wedge$
$\forall m \in \text{movie}\ [m(\text{director}) = \text{“Berto”} \ \wedge \ \exists t \in \text{movie}\ (m(\text{title}) =$
$t(\text{title}) \wedge t(\text{actor}) = y(\text{actor}))]]\}$

A: YES B: NO

# Examples (Movie Database)

- Find actors playing in every movie by Berto

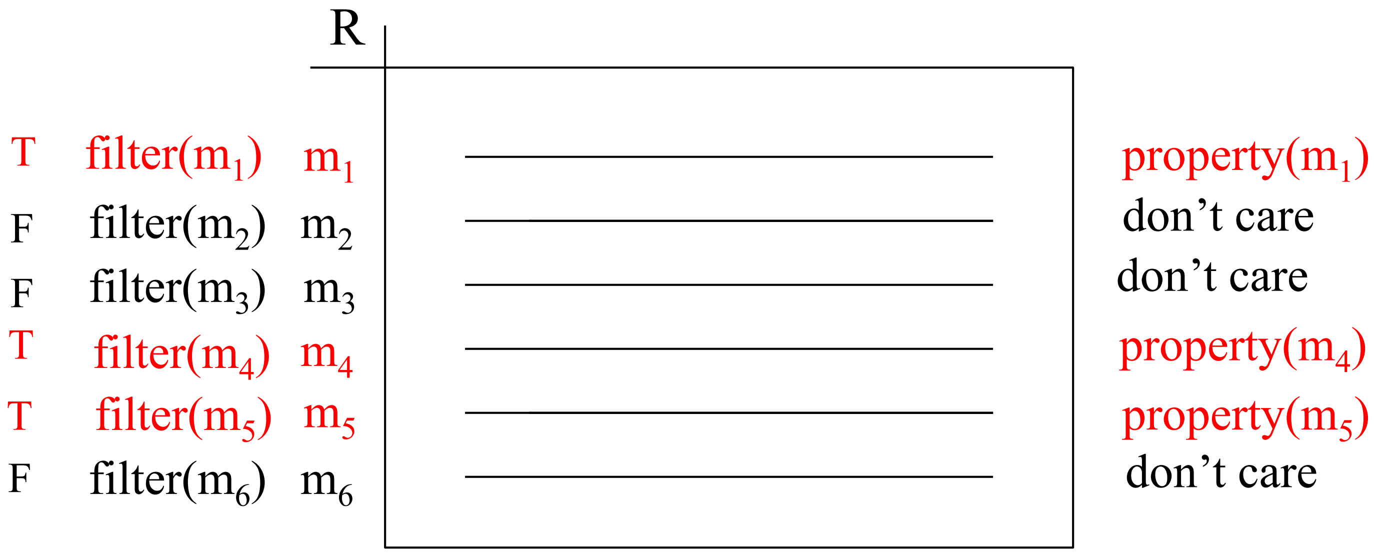
$\{a\text{: actor} \mid \exists y \in \text{movie}\ [a(\text{actor}) = y(\text{actor}) \ \wedge$
$\forall m \in \text{movie}\ [m(\text{director}) = \text{“Berto”} \rightarrow \exists t \in \text{movie}\ (m(\text{title}) =$
$t(\text{title}) \wedge t(\text{actor}) = y(\text{actor}))]]\}$

Typical use of $\forall$:

$$\forall\ \mathbf{m} \in R\ [\ \text{filter}(\mathbf{m}) \rightarrow\ \text{property}(\mathbf{m})]$$

Intuition: check property(**m**) for those **m** that satisfy filter(**m**)
we don't care about the **m**'s that do not satisfy filter(**m**)

$$\forall\, \mathbf{m} \in R\ [\ \mathrm{filter}(\mathbf{m}) \rightarrow \mathrm{property}(\mathbf{m})]$$



$$\forall\, \mathbf{m} \in \text{movie}\ [\ \text{filter}(\mathbf{m}) \rightarrow \text{property}(\mathbf{m})]$$

$$\downarrow$$

$$m.Dir = Berto$$

| | | Movie | title | director | actor | |
|---|---|---|---|---|---|---|
| T | filter($m_1$) | $m_1$ | | Berto | | property($m_1$) |
| F | filter($m_2$) | $m_2$ | | Hitchcock | | don’t care |
| F | filter($m_3$) | $m_3$ | | Hitchcock | | don’t care |
| T | filter($m_4$) | $m_4$ | | Berto | | property($m_4$) |
| T | filter($m_5$) | $m_5$ | | Berto | | property($m_5$) |
| F | filter($m_6$) | $m_6$ | | Fellini | | don’t care |

# Tuple Calculus and SQL

- Example: “Find theaters showing movies by Bertolucci”:
  - SQL:
    - SELECT s.theater
      FROM schedule s, movie m
      WHERE s.title = m.title  AND m.director = “Bertolucci”
  - tuple calculus:
    - { t: theater | $\exists$ s $\in$ schedule $\exists$ m $\in$ movie [ t(theater) = s(theater) $\wedge$ s(title) = m(title) $\wedge$ m(director) = Bertolucci ] }

# Basic SQL Query

SELECT $A_1, \ldots, A_n$
FROM $R_1, \ldots, R_k$
WHERE $cond(R_1, \ldots, R_k)$

for each tuple $r_1$ in $R_1$
  for each tuple $r_2$ in $R_2$
    …….
      for each tuple $r_m$ in $R_m$

if condition($r_1, r_2, \ldots, r_m$) then output in answer attributes $a_1, \ldots, a_n$ of $r_1, \ldots, r_m$

# Tuple Calculus

$$\{t: A_1, \ldots, A_n \mid \exists r_1 \in R_1 \ldots \exists r_k \in R_k [\wedge_j\, t(A_j) = r_{ij}(A_j) \wedge cond(r_1, \ldots, r_k)]\}$$

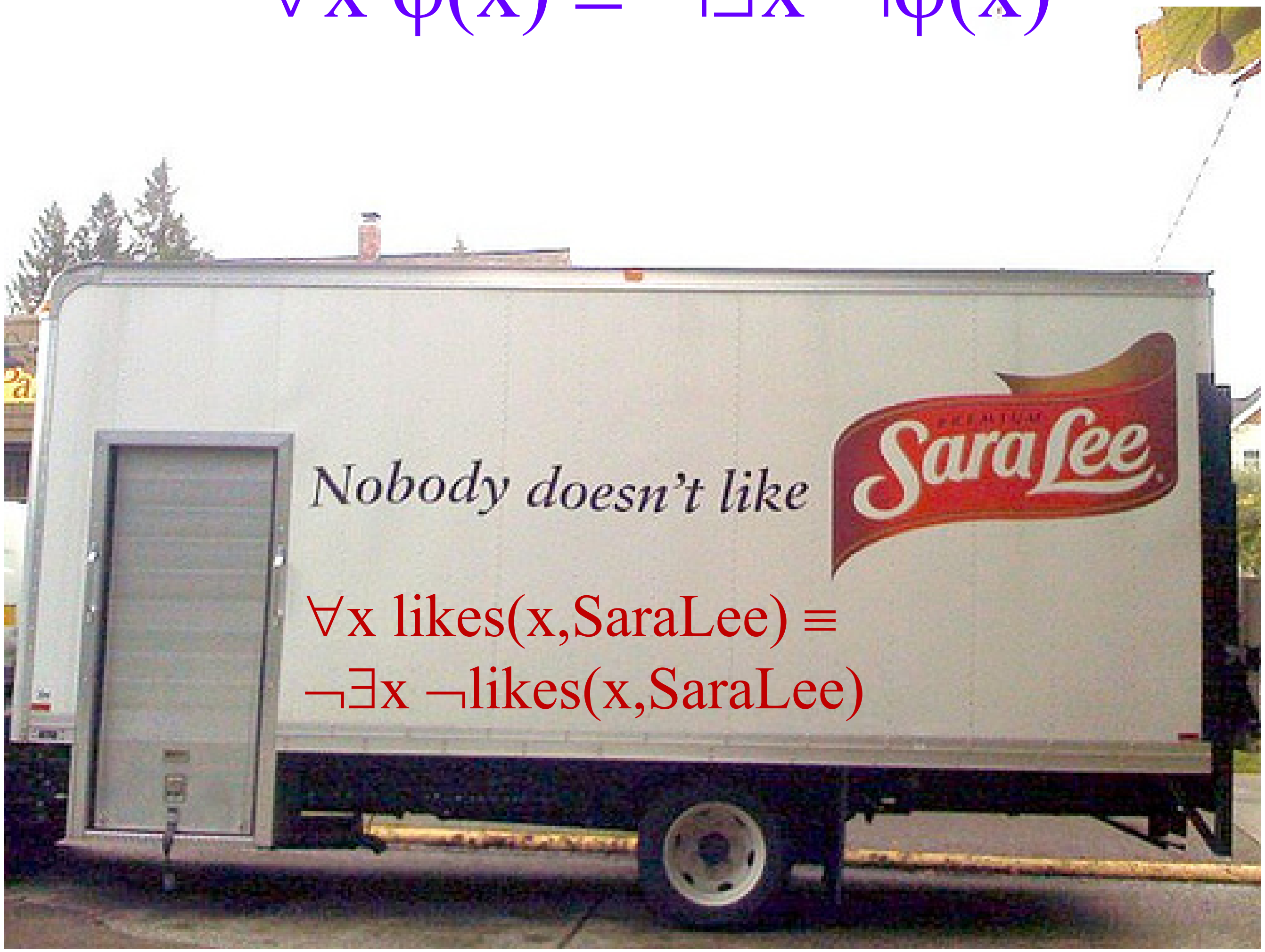
- Note: basic SQL query uses only $\exists$;
- no explicit construct for $\forall$

# Using Tuple Calculus to Formulate SQL Queries

- Example: "Find actors playing in every movie by Berto"
- Tuple calculus
  - $\{a: actor \mid \exists y \in movie\ [a(actor) = y(actor)\ \wedge$
    $\forall m \in movie\ [m(director) = \text{"Berto"} \rightarrow \exists t \in movie\ (m(title) =$
    $t(title) \wedge t(actor) = y(actor))]]\}$
- Eliminate $\forall$:
  - $\{a: actor \mid \exists y \in movie\ [a(actor) = y(actor)\ \wedge$
    $\neg\exists m \in movie\ [m(dir) = \text{"Berto"} \wedge \neg\exists t \in movie\ (m(title) = t(title)$
    $\wedge\ t(actor) = y(actor))]]\}$
- Rule: $\forall x \in R\ \varphi(x) \equiv \neg\exists x \in R\ \neg\varphi(x)$

"every x in R satisfies φ(x) iff
there is no x in R that violates φ(x)"

$$\forall x\ \varphi(x) \equiv \neg\exists x\ \neg\varphi(x)$$




$$\forall x\ \text{likes}(x,\text{SaraLee}) \equiv \neg\exists x\ \neg\text{likes}(x,\text{SaraLee})$$

# Convert to SQL query

- Basic rule: one level of nesting for each "$\neg\exists$"

$$\{a: actor \mid \exists y \in movie\ [a(actor) = y(actor) \wedge \neg\exists m \in movie\ [m(dir) = \text{"Berto"} \wedge \neg\exists t \in movie\ (m(title) = t(title) \wedge t(actor) = y(actor))]]\}$$

```
SELECT y.actor  FROM movie y
WHERE NOT EXISTS
(SELECT * FROM movie  m
WHERE m.dir = 'Berto' AND
NOT EXISTS
        (SELECT *
        FROM movie t
        WHERE m.title = t.title  AND t.actor = y.actor ))
```

# Another possibility (with similar nesting structure)

```
SELECT actor  FROM movie
WHERE actor  NOT IN
(SELECT s.actor
FROM movie s, movie m
WHERE  m.dir  = 'Berto'
AND s.actor  NOT IN
        (SELECT t.actor
        FROM movie t
        WHERE m.title = t.title ))
```

- Note: Calculus is more flexible than SQL because of the ability to mix ∃ and ∀ quantifiers

# Examples

Beer drinker's database:

| frequents | drinker | bar |
|---|---|---|

| serves | bar | beer |
|---|---|---|

| likes | drinker | beer |
|---|---|---|

Find the drinkers who frequent some bars serving Coors

| frequents | drinker | bar |
| --- | --- | --- |
| | | |

| serves | bar | beer |
| --- | --- | --- |
| | | |

| likes | drinker | beer |
| --- | --- | --- |
| | | |

| answer | drinker |
| --- | --- |
| | |

Find the drinkers who frequent at least one bar serving a beer they like

| frequents | drinker | bar |
|---|---|---|
| | | |

| serves | bar | beer |
|---|---|---|
| | | |

| likes | drinker | beer |
|---|---|---|
| | | |

| answer | drinker |
|---|---|
| | |

Find the drinkers who frequent ONLY bars serving a beer they like

| frequents | drinker | bar |
|---|---|---|
| | | |

| serves | bar | beer |
|---|---|---|
| | | |

| likes | drinker | beer |
|---|---|---|
| | | |

| answer | drinker |
|---|---|
| | |