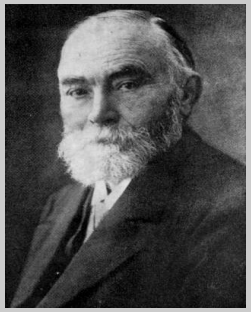
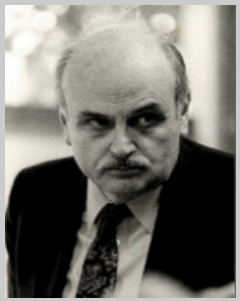
Relational DB: The Origins



Frege: FO logic

Tarski: Algebra for FO



Codd: Relational databases

relational calculus

Relational Calculus (aka FO)

- Models data manipulation core of SQL
  Idea: specify "what" not "how"
- General form:
  {t | property (t)}
- property (t) is described by a language based on predicate calculus (first-order logic)

Relational Calculus Example

Display the movie table

In SQL

```
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": $\mathbf{m \in R}$

Relational Calculus Example

Find the directors and actors of currently playing movies

In SQL

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 s.t.
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

Relational Calculus Example

Find the directors and actors of currently playing movies

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

In logic notation (tuple relational calculus)

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

Quantifiers

∃ m ∈ R: Existential quantification
"there exists some tuple m in relation R …."

Sometimes need to say:
"for every tuple m …."

e.g., "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"

∀ m ∈ movie ∃ t ∈ movie [m(Director) = t(Actor)]

(The answer to this query is true or false)

∀ m ∈ R: Universal quantification
"for every tuple m in relation R …."

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

- 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: $\forall t \in R\ \varphi(t)$

  where $\varphi(t)$ a formula in which t not quantified (it is “free”)

More on quantifiers

- Scope of quantifier:
  scope of $\exists t \in R\ \varphi(t)$ is φ
  scope of $\forall t \in R\ \varphi(t)$ is φ
- Free variable:
  not in scope of any quantifier
  free variables are the “parameters” of the formula
- Rule: in quantification $\exists t \in R\ \varphi(t)$, $\forall t \in R\ \varphi(t)$
  t must be free in φ

Quantifier Examples

{ t: Director, Actor | ∃ m ∈ movie ∃ s ∈ schedule
[t(Director) = m(Director) ∧ t(Actor) = m(Actor) ∧ m(Title) = s(Title)] }

[t(Director) = m(Director) ∧ t(Actor) = m(Actor) ∧ m(Title) = s(Title)]

free: t, m, s

∃ s ∈ schedule

[t(Director) = m(Director) ∧ t(Actor) = m(Actor) ∧ m(Title) = s(Title)]

free: t, m

∃ m ∈ movie ∃ s ∈ schedule

[t(Director) = m(Director) ∧ t(Actor) = m(Actor) ∧ m(Title) = s(Title)]

free: t

Example in predicate logic

A statement about numbers:

$\exists x \forall y \forall z [x = y * z \longrightarrow ((y = 1) \vee (z = 1))]$

"there exists at least one prime number x"

A "query" on numbers:

$\varphi(x)$: $\forall y \forall z [x = y * z \longrightarrow ((y = 1) \vee (z = 1))]$

This defines the set $\{x \mid \varphi(x)\}$ of prime numbers.
It consists of all x that make $\varphi(x)$ true.

Semantics of Tuple Calculus

- Active domain:

  A set of values in the database, or mentioned in the query result.
  Tuple variables range over the active domain

- Note:

  A query without free variables always evaluates to true or false

  e.g., “Sky is by Berto” is expressed without free variables:

  $\exists m \in movie\ [m(title) = \text{“Sky”} \wedge m(director) = \text{“Berto”}]$

  This statement is true or false

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 in active domain with make φ(v) true

Note:

φ(v) has no free variables so it evaluates to true or false

Movie Examples Revisited

Find titles of currently playing movies

```
select Title
from Schedule
```

Find the titles of all movies by “Berto”

```
select Title
from Movie
where Director=“Berto”
```

Find the titles and the directors of all currently playing movies

```
select Movie.Title, Director
from Movie, Schedule
where Movie.Title = Schedule.Title
```

Movie Examples Revisited

Find titles of currently playing movies

{t: title | ∃s ∈schedule [s(title) = t(title)]}

Find the titles of all movies by "Berto"

{t: title| ∃m ∈ movie [m(director) = "Berto" ∧ t(title) = m(title)]}

Find the titles and the directors of all currently playing movies

{t: title, director | ∃s ∈schedule ∃m ∈ movie
[s(title) = m(title) ∧ t(title) = m(title) ∧ t(director) = m(director)]}

Movie Examples Revisited

- Find actors playing in every movie by Berto

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

Is the following correct?

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

A: YES B: ~~NO~~

Movie Examples Revisited

- Find actors playing in every movie by Berto

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

Typical use of $\forall$:

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

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**)

Movie Examples Revisited

- Find actors playing in every movie by Berto

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

Is this correct?

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

A: YES B: NO

Movie Examples Revisited

Is this correct?

{a: actor | ∃y ∈ movie [a(actor) = y(actor) ∧
∀m ∈ movie **∃t ∈ movie** [m(director) = "Berto" → (m(title) =
t(title) ∧ t(actor) = y(actor))]]}

A: ~~YES~~ B: NO

∃t (φ ∨ ψ) = **∃t** φ ∨ **∃t** ψ
∃t φ = φ if **t** does not occur in φ

Is the following correct:
∃t (φ ∧ ψ) = **∃t** φ ∧ **∃t** ψ

A: YES B: NO

Movie Examples Revisited

Correct:

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

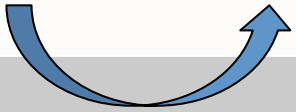
$\exists t\ (\varphi \vee \psi) = \exists t\ \varphi \vee \exists t\ \psi$
$\exists t\ \varphi = \varphi$ if t does not occur in φ

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

Movie Examples Revisited

Correct:

{a: actor | ∃y ∈ movie [a(actor) = y(actor) ∧
∀m ∈ movie **∃t ∈ movie** [m(director) = "Berto" → (m(title) =
t(title) ∧ t(actor) = y(actor))]]}



Is this also correct (can we switch ∀ and ∃)?

{a: actor | ∃y ∈ movie [a(actor) = y(actor) ∧
∃t ∈ movie ∀m ∈ movie [m(director) = "Berto" → (m(title) =
t(title) ∧ t(actor) = y(actor))]]}

A: YES B: ~~NO~~

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:

  $\{ t: theater \mid \exists s \in schedule \; \exists m \in movie \; [t(theater) = s(theater) \wedge s(title) = m(title) \wedge m(director) = Bertolucci \;] \}$

Basic SQL Query

SQL

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

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

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

- Eliminate ∀:

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

- Rule: $\forall x \in R\ \varphi(x) \equiv \neg\exists x \in R\ \neg\varphi(x)$

  “every x in R satisfies $\varphi(x)$ iff
  there is no x in R that violates $\varphi(x)$”

Convert to SQL query

- Basic rule: one level of nesting for each "¬∃"

{a: actor | ∃y ∈ movie [a(actor) = y(actor) ∧
¬∃m ∈ movie [m(dir) = "Berto" ∧ ¬∃t ∈ movie (m(title) = t(title)
∧ 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

Calculus Vs. Algebra

- Theorem: Calculus and Algebra are equivalent
- Basic Correspondence:

Example

- "Find theaters showing movies by Bertolucci":
  SQL:
  - **SELECT** s.theater
    FROM schedule s, movie m
    WHERE s.title = m.title **AND** m.director = 'Berto'

  tuple calculus:
  - { t: theater | ∃ s ∈ schedule ∃ m ∈ movie [t(theater) = s(theater) ∧ s(title) = m(title) ∧ m(director) = Berto] }

  relational algebra:

$$\pi_{theater}\,(schedule \bowtie \sigma_{dir\,=\,Berto}\,(movie))$$

Note: number of items in FROM clause = (number of joins + 1)