Relational Data Base

Data base

A collection of files storing related data

DBMS

A big program written by someone else that allows us to manage efficiently a large database and allows it to persist over long periods of time

What a DBMS does?

Describe real-world entities in terms of stored data

Persistently store large datasets

Efficiently query & update

Change structure

Data model

Mathematical formalism for describing the data

Instance: the actual data

Schema: describe what data is being stored

Query language: how to retrieve and manipulate data

Relation

A set of tuples: each tuple appears 0 or 1 times in the table

Order of the rows is unspecified

Degree (arity) of a relation

Number of attributes

Keys

One or multiple attributes that uniquely identify a record (a relation can have many keys but only one can be chosen as primary key)

Foreign key

Attributes whose value is a key of a record in some other relation (also called semantic pointer)

Physical data independence

The logical definition of the data remains unchanged, even when we make changes to the actual implementation

First Normal Form

All relations must be flat

Inner join

Each row in the result must come from both tables in the join

Outer join

Include rows from only one of the two tables

```
tableA (LEFT/RIGHT/FULL) OUTER JOIN tableB ON p
```

- Left outer join:
  - Include tuples from `tableA` even if no match
- Right outer join:
  - Include tuples from `tableB` even if no match
- Full outer join:
  - Include tuples from both even if no match

- In all cases:
  - Patch tuples without matches using `NULL`

Including empty groups

In the result of a group by query, there is one row per group in the result

Count(*) is never 0

```
SELECT x.manufacturer, count(*)
FROM Product x, Purchase y
WHERE x.pname = y.product
GROUP BY x.manufacturer
```

Subqueries

A subquery is a SQL query nested inside a larger query

A subquery may occur in: SELECT, FROM or WHERE

Monotone

Whenever we add tuples to one or more input tables, the answer to the query will not lose any of the tuples

If Q is a SELECT-FROM-WHERE query that doesn't have subqueries or aggregates, then it is monotone

If a query is not monotonic, then we cannot write it as a SELECT-FROM-WHERE query without nested subqueries

Process sequence

From -> Where -> Group -> Having -> Order by -> Select

Everything in SELECT must be either a group-by attribute, or an aggregation!!!!!

Relational algebra

An RA expression is called a query plan

SQL vs. RA

SQL is a declarative language where we say what data we want to retrieve

RA is an algebra where we say how we want to retrieve the data

Join

Theta-join: $R \bowtie_\theta S = \sigma_\theta (R \times S)$

- Join of R and S with a join condition θ
- Cross-product followed by selection θ
- No projection

Equijoin: $R \bowtie_\theta S = \sigma_\theta (R \times S)$

- Join condition θ consists only of equalities
- No projection

Natural join: $R \bowtie S = \pi_A (\sigma_\theta (R \times S))$

- Equality on **all** fields with same name in R and in S
- Projection π_A drops all redundant attributes

Datalog

A query language for relational model

Extensional vs. Intensional Database Predicates (EDB, IDB)

Extensional Database Predicates Relations are stored in the database whereas Intensional Database Predicates Relation are defined by one or more rules

A predicate must be either IDB or EDB, not both. Thus, an IDB predicate can appear in the body or head of a rule whereas EDB only in the body

Facts = tuples in the database

```
Actor(344759,'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

Rules = queries

```
Q1(y) :- Movie(x,y,z), z=1940.
```

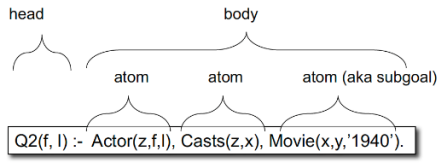
```
Q2(f,l) :- Actor(z,f,l), Casts(z,x),
           Movie(x,y,1940).
```

```
Q3(f,l) :- Actor(z,f,l), Casts(z,x1), Movie(x1,y1,1910),
           Casts(z,x2), Movie(x2,y2,1940).
```

Extensional Database Predicates = EDB = Actor, Casts, Movie
Intensional Database Predicates = IDB = Q1, Q2, Q3

Facts: tuples in the database **Rules:** queries

Terminology



f, l = head variables
x,y,z= existential variables

```
Q(args) :- R1(args), R2(args), ....
```

$R_i(args_i)$ called an *atom*, or a *relational predicate*

$R_i(args_i)$ evaluates to true when relation R_i contains the tuple described by $args_i$.

- Example: Actor(344759, 'Douglas', 'Fowley') is true

In addition we can also have arithmetic predicates

- Example: z > '1940'.

Book uses AND instead of ,

```
Q(args) :- R1(args) AND R2(args) ....
```

Safe Datalog rules

A Datalog rule is safe if every variable appears in some positive relational atom

wildcard in negation is NEVER safe as each wildcard is treated as unique, even if there are multiple

Stratified Datalog

A Datalog program is stratified if it can be partitioned into strata. Only IDB predicates defined in strata 1,2,…,n may appear under ! or aggregate in stratum n+1

```
D(x,y) :- ParentChild(x,y).
D(x,z) :- D(x,y), ParentChild(y,z).
```

Stratum 1

```
T(p,c) :- D(p,_), c = count : { D(p,y) }.
Q(d) :- T(p,d), p = "Alice".
```

Stratum 2

May use D in an agg since it was defined in previous stratum