COL 362 & COL 632

Relational Algebra and SQL

20 Jan 2023

Basic operations

- Selection σ (choose subset of rows)
- Projection Π (choose subset of columns)
- Cross product $\times$
- Union $\cup$
- Difference $-$
- Rename ρ
- Join $\bowtie$

Same operations as those on any sets
Apply operations on tuples with same schema

Cross product

R1.Name , R2.Name

|R1| = x

|R2| = y

R3

x * y

R1 = R2

$$R3 = R1 \times R2$$

Actors

Name	Age	Addr
Priyanka Chopra	38	Mumbai
Anthony Hopkins	81	LA
Bill Nighy	69	LA
Abhishek Bachchan	45	Mumbai

Movies

Name	Year	Title
Priyanka Chopra	2011	Don-II
Anthony Hopkins	2011	MI-IV
Bill Nighy	2009	Valkyrie
Abhishek Bachchan	2010	Raavan

Actor.name	Age	Addr	Movies.Name	Year	Title
Priyanka Chopra	38	Mumbai	Priyanka Chopra	2011	Don-II
…15 more rows…					

Joins (1/2)

$R1 \bowtie R_2 \subseteq R1 \times R_2$

Actors $\bowtie$ Movies

$$R3 = R1 \bowtie_C R2$$

C is a condition on attributes of R1 and/or R2

Actors

Name	Age	Addr
PC	38	Mumbai
AH	81	LA
BN	69	LA
AB	45	Mumbai

Movies

Name	Year	Title
PC	2011	Don-II
AH	2011	Thor: R
BN	2009	Valkyrie
AB	2010	Raavan

Return all information about actors and their movies

$$Actors \bowtie_{A.Name=M.Name} Movies$$

Name	Age	Addr	Year	Title
PC	38	Mumbai	2011	Don-II
AH	81	LA	2011	Thor: R
BN	69	LA	2009	Valkyri e
AB	45	Mumbai	2010	Raavan

Joins (2/2)

- Natural joins
  - implicitly compares **attributes of the same name** for equality
- Theta join
  - conditions not restricted to equality
- Left-outer/right-outer/full-outer joins
  - non-matching tuples are still returned
- Self-join
  - table joining with itself

Left outer joins

$$R = X ⟕ Y$$

Actors

Name	Age	Addr
PC	38	Mumbai
AH	81	LA
BN	69	LA
AB	45	Mumbai

Movies

Name	Year	Title
PC	2011	Don-II
AH	2011	Thor: R
AB	2010	Raavan

Return all information about actors and their movies

$$Actors ⟕_{A.Name=M.Name} Movies$$

Name	Age	Addr	Year	Title
PC	38	Mumbai	2011	Don-II
AH	81	LA	2011	Thor: R
BN	69	LA	null	null
AB	45	Mumbai	2010	Raavan

Self Join

Return all grandparents and their grand children

Actors

Name	Age	Addr	Parent
PC	38	Mumbai	Madhu
AH	81	LA	Muriel
BN	69	LA	Catherine
AB	45	Mumbai	Jaya
Jaya	63	Mumbai	Indira

Actors_1

Name	Age	Addr	Parent
PC	38	Mumbai	Madhu
AH	81	LA	Muriel
BN	69	LA	Catherine
AB	45	Mumbai	Jaya
Jaya	63	Mumbai	Indira

Actors ⋈ Actors_1
A.Parent = A1.Name

SQL - 1

SQL Query Language

- SQL query language is **nonprocedural (declarative)**. A query takes as input several tables (possibly only one) and always returns a single table.
- Example to find all instructors in Comp. Sci. dept

  select *name*
  from *instructor*
  where *dept_name* = 'Comp. Sci.'

- SQL is often embedded in some higher-level language
- Application programs generally access databases through one of
  - Language extensions to allow **embedded SQL**
  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

SQL Parts

- DML – Data Manipulation Language provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database.
- DDL – Data Definition Language
- **Integrity** – the DDL includes commands for specifying integrity constraints.
- **View definition** -- The DDL includes commands for defining views.
- **Transaction control** –includes commands for specifying the beginning and ending of transactions.
- **Embedded SQL and dynamic SQL** -- define how SQL statements can be embedded within general-purpose programming languages.
- **Authorization** – includes commands for specifying access rights to relations and views.

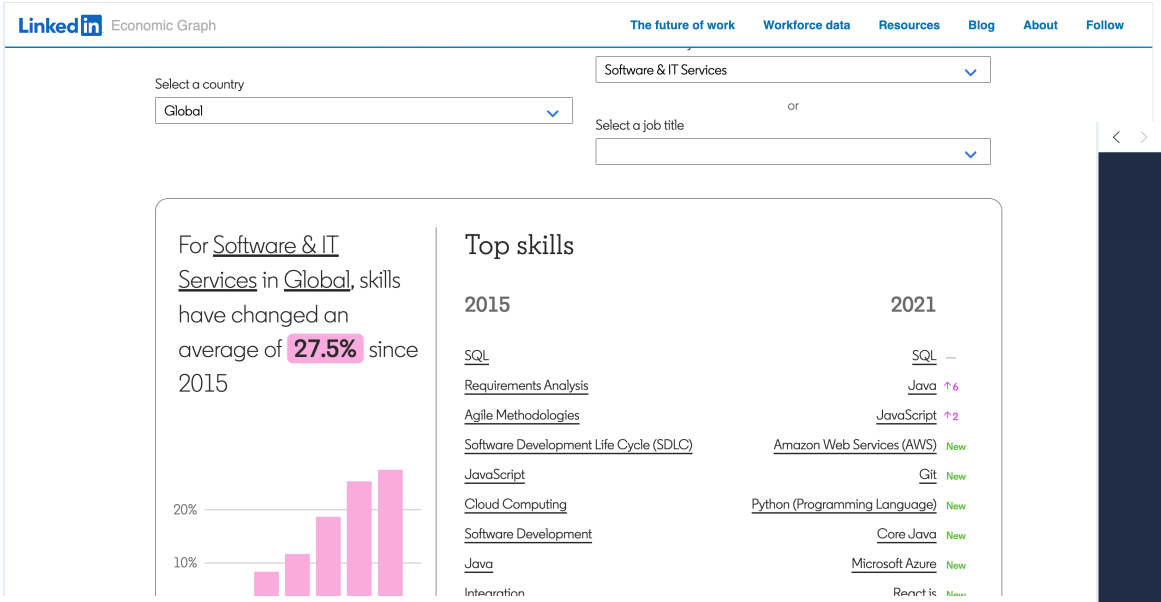
Data Definition Language (DDL)

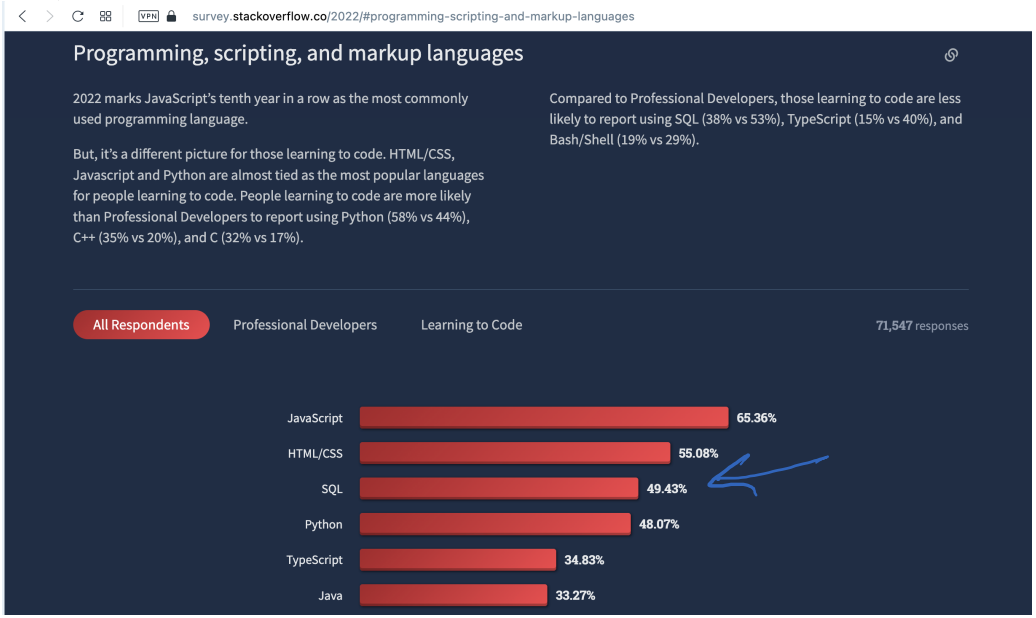
- Specification notation for defining the database schema
- Example: **create table** instructor (
  ID **char**(5),
  name **varchar**(20),
  dept_name **varchar**(20),
  salary **numeric**(8,2))
- DDL compiler generates a set of table templates stored in a data dictionary
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
  - Primary key (ID uniquely identifies instructors)
  - Authorization
  - Who can access what

Data Manipulation Language (DML)

- Language for accessing and updating the data organized by the appropriate data model
  - DML also known as **query language**
- There are basically two types of data-manipulation language
  - **Procedural DML** -- require a user to specify what data are needed and how to get those data.
  - **Declarative DML** -- require a user to specify what data are needed without specifying how to get those data.
- Declarative DMLs are usually easier to learn and use than are procedural DMLs.
- Declarative DMLs are also referred to as non-procedural DMLs
- The portion of a DML that involves information retrieval is called a **query** language.

SQL still the top-dog !!





SQL – basic structure

```
SELECT L
FROM R
WHERE C
```

Attributes of the output relation

List of all relations involved

Actors

Conditions to be satisfied

Name, Age

AH 81
BN 69

```
SELECT Name, ~~DOB~~ Age
FROM Actors
WHERE Actors.Address = 'LA';
```

```
SELECT *
FROM Actors
WHERE Actors.Address = 'LA';
```

Return all actors living in LA

$$\sigma_{Address='LA'}(Actors)$$

AH, 81, LA
BN, 81, LA

Name	Age	Addr
Priyanka Chopra	38	Mumbai
Anthony Hopkins	81	LA
Bill Nighy	69	LA
Abhishek Bachchan	45	Mumbai

Selection

Return all actors whose age is more than 35.

$$\sigma_{Age>35}(Actors)$$

```
SELECT *
FROM Actors
WHERE Age > 35
```

Return all actors whose age is more than 35 and who live in LA

$$\sigma_{Age>35 \text{ and } Address='LA'}(Actors)$$

```
SELECT *
FROM Actors
WHERE Age > 35 AND
      Addr = 'Mumbai'
```

Projection

Actors

Name	Age	Addr
Priyanka Chopra	38	Mumbai
Anthony Hopkins	81	LA
Bill Nighy	69	LA
Abhishek Bachchan	45	Mumbai

Return the name and age of all actors

$$\Pi_{Name,Age}(Actors)$$

```
SELECT Name, Age
FROM Actors
```

Return the addresses of the actors

$$\Pi_{Addr}(Actors)$$

```
SELECT Addr
FROM Actors
```

```
SELECT DISTINCT (Addr)
FROM Actors
```

$|A \cup B| = |A| + |B| - \cancel{|A \cap B|}$

Sets = unique elements

Relational algebra for bags

Bags = repeats are allowed.

- Efficiency issues if we consider relations as sets
  - Extra effort to eliminate duplicates
- Select, project, join (SPJ) work exactly the same
  - Applied to one tuple at a time
- **Set operations become bag operations**
  - Need to be careful about semantics
  - Union, Intersection, Difference

Bag $\{a, b, a, a\}$

$\Downarrow$

Set $\{a, b\}$

$B_1 = \{a, b, a, a\}$

$B_2 = \{b, b, c, a\}$

$B_1 \cup_{Bag} B_2 = \{a, a, a, a, b, b, b, c\}$

Equi-Joins

JOIN
OUTER JOIN

Actors

Name	Age	Addr
PC	38	Mumbai
AH	81	LA
BN	69	LA
AB	45	Mumbai

Movies

Name	Year	Title
PC	2011	Don-II
AH	2011	Thor: R
BN	2009	Valkyrie
AB	2010	Raavan

Age

Return all information about actors and their movies

$$Actors \bowtie_{A.Name=M.Name} Movies$$

```
SELECT *
FROM Actors, Movies
WHERE Actors.Name = Movies.Name
```

Actors. Name, Ac, Age, Act. Addr,

. . . .

Left outer joins

Return all information about actors and their movies

$$Actors \bowtie_{A.Name=M.Name} Movies$$

Actors

Name	Age	Addr
PC	38	Mumbai
AH	81	LA
BN	69	LA
AB	45	Mumbai

Movies

Name	Year	Title
PC	2011	Don-II
AH	2011	Thor: R
AB	2010	Raavan

```
SELECT *
FROM Actors LEFT OUTER JOIN Movies
ON (Actors.Name = Movies.Name)
```

What happens when you compare something with a null value? Or when you compare a null with a null?

Self Join

Return all grandparents and their grand children

Actors

Name	Age	Addr	Parent
PC	38	Mumbai	Madhu
AH	81	LA	Muriel
BN	69	LA	Catherine
AB	45	Mumbai	Jaya
Jaya	63	Mumbai	Indira

Actors_1

Name	Age	Addr	Parent
PC	38	Mumbai	Madhu
AH	81	LA	Muriel
BN	69	LA	Catherine
AB	45	Mumbai	Jaya
Jaya	63	Mumbai	Indira

```
SELECT *
FROM Actors AS Actors1, Actors AS Actors2
WHERE Actors1.Parent = Actors2.Name
```