Why DB?

- Alternative → Text File / CSV

Uses of DB software

- Provides simple and easy to use language called SQL
- Faster → Uses indexing to retreive data
- Reliable
- Secure

(Even if hardware/hard-disk crashes, DB's have backup to provide data access)

- Uses triplecate storage - stores a copy of data in different hard-disk

Types of DB's

- Relational DB's → Data is stored on multiple tables
  - Oracle
  - MySQL
  - SQLServer
- Non-relational DB's → Data is stored in the form Dictionary (JSON)
  - MongoDB
  - NoSQL

SQL

1. **DML** → Data Manipulation Language
   - SELECT
   - INSERT
   - UPDATE
   - DELETE
2. **DDL** → Data Definition Language
   - TRUNCATE → deletes the contents of the table (not the whole table)
   - NOT NULL
   - UNIQUE
   - PRIMARY KEY
   - FOREIGN KEY
   - CHECK
   - DEFAULT
   - INDEX
   - CREATE
   - ALTER
     - ADD
     - MODEIFY
     - DROP → deletes the whole table
3. **DCL** → Data Control Language (ensures data safety)
   - Mainly used by the DB admins. They have the right to modify or create or delete data in the database, but not users.
   - GRANT
   - REVOKE
   - Admins → responsible for well running of the database (services).

Structured Quer Language → Developed by IBM in the year 1970 by Researchers

- Widely used in relational databases
- It is a standard way to query/obtain/add/delete/modify the data at any point of time
- Not a general purpose programming language
  - Domain specific language in the domain of databases

- In procedural/general programming language, programmer gives step-by-step instructions.
- SQL is a declarative/domain specific language, programmer tells what she/he needs and how to get that is someother's problem.

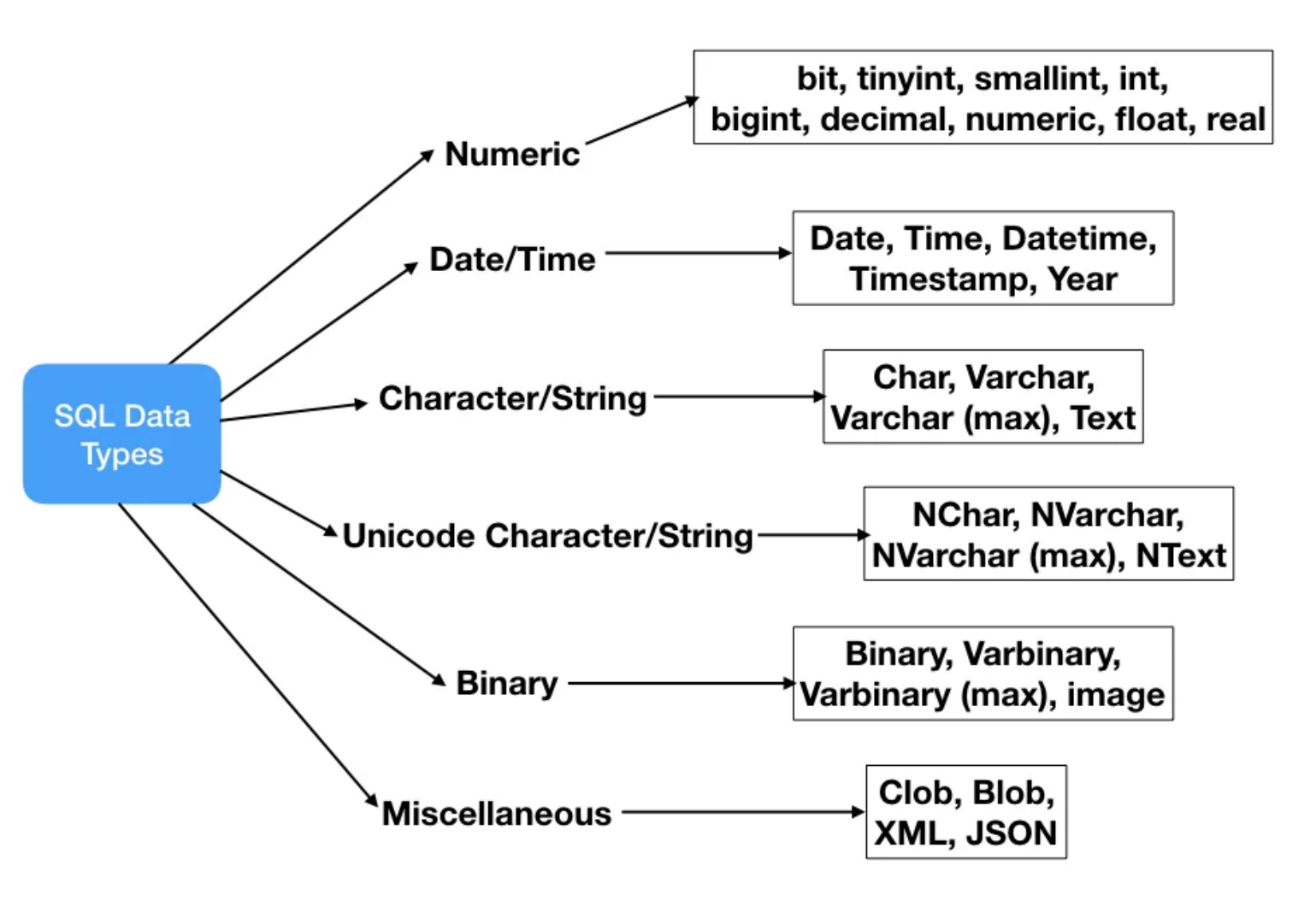
Command Execution

- Parser/Compiler → tries to understand what is the command that is given
  - generates the code in either c/c++/java/python
  - responsible to show errors if there persist some errors
- Query Optimizer → finds the optimal way to execute the command/query
- Query Executor → executes the code on the db and display the entries

Initial steps

- USE ; → to use the database
- SHOW TABLES; → lists all the tables which are there in the database
- DESCRIBE → describes a particular table selected

Data types



Credits - Image from Internet

ResultSet

ResultSet is set of rows and columns.

- It can be whole data table or subset of it.
- Output generated from a SELECT query.
- Query that is given to select a subset is always faster than selecting the whole data able as it is.
- Row order is preserved in SQL queries.

OFFSET

The main advantage of using OFFSET is to ignore the first `<specified_number>`.

Wild card characters

- `%` → all characters
- `_` → exactly one character
- `\` → escape character

Aggregate functions

- COUNT
- MIN
- MAX
- SUM
- AVG

Important commands

DISTINCT

Command

```
> SELECT DISTINCT genre FROM movies_genres;
```

Output

genre
Documentary
Short
Comedy
...
Adventure
Film-Noir

GROUP BY & ORDER BY (together)

Command

```
> SELECT genre, AVG(prob), COUNT(genre)
-> FROM directors_genres
-> GROUP BY genre;
```

Output

genre	AVG(prob)	COUNT(genre)
Short	0.8261636824405604	28294
Drama	0.7055499588868819	25357
...	...	...
Music	0.5782766784248387	2472
Film-Noir	0.11276367777963077	204

Command

```
> SELECT genre, ROUND(AVG(prob), 3), COUNT(genre)
-> FROM directors_genres
-> GROUP BY genre;
```

Output

genre	ROUND(AVG(prob), 3)	COUNT(genre)
Short	0.826	28294
Drama	0.706	25357
...	...	...
Music	0.578	2472
Film-Noir	0.113	204

Command

```
> SELECT year, COUNT(year)
-> FROM movies
-> GROUP BY year
-> ORDER BY year ASC;
```

Output

year	COUNT(year)
1888	2
1890	3
...	...
2007	7
2008	1

Command

```
> SELECT genre, ROUND(AVG(prob), 3) AS p, COUNT(genre)
-> FROM directors_genres
-> GROUP BY genre
-> ORDER BY p;
```

Output

genre	p	COUNT(genre)
Film-Noir	0.113	204
War	0.326	2931
...	...	...
Short	0.826	28294
Animation	0.863	4417

HAVING & GROUP BY (together)

- `HAVING` is typically used along with `GROUP BY`.
- `HAVING` is same as `WHERE` (when not using `GROUP BY`).
- `WHERE` is applied on individual rows and `HAVING` is applied on groups.

Command

```
> SELECT year, COUNT(year) AS yc
-> FROM movies
-> GROUP BY year
-> HAVING yc >= 1000;
```

Output

year	yc
2002	12056
2000	11643
...	...
1953	2549
1926	2137

Command

```
> SELECT year, COUNT(year) as yc
-> FROM movies
-> GROUP BY year
-> HAVING yc >= 1000
-> ORDER BY year;
```

Output

year	yc
1898	1004
1910	1276
...	...
2004	8718
2005	1449

Command

```
> SELECT year, COUNT(year) AS yc
-> FROM movies
-> WHERE rankscore > 9.6
-> GROUP BY year
-> ORDER BY yc;
```

Output

year	yc
1930	1
1962	1
...	...
2004	18
2003	26

Command

```
> SELECT year, COUNT(year) AS yc, rankscore
-> FROM movies
-> WHERE rankscore > 9.6
-> GROUP BY year
-> HAVING yc > 5
-> ORDER BY yc;
```

Output

year	yc	rankscore
1987	6	9.6
1979	6	9.6
...	...	...
2002	18	9.8
2003	26	9.8

JOINs (joining two tables where particular condition is TRUE)

- By default `JOIN` refers to `INNER JOIN`.

Command

```
> SELECT m.name, g.genre
-> FROM movies AS m
-> JOIN movies_genres AS g
-> ON m.id = g.movie_id
-> LIMIT 20
```

Output

name	genre
#7 Train: An Immigrant Journey, The	Documentary
#7 Train: An Immigrant Journey, The	Short
...	...
$ 2500 Bride, The	Drama
$ 2500 Bride, The	Romance

Command

```
> SELECT * FROM movies AS m
-> JOIN movies_genres AS g
-> ON m.id = g.movie_id
-> LIMIT 20;
```

Output

id	name	year	rankscore	movie_id	genre
1	#7 Train: An Immigrant Journey, The	2000	NULL	1	Documentary
1	#7 Train: An Immigrant Journey, The	2000	NULL	1	Short
...	...	...	...	...	...
14	$ 2500 Bride, The	1912	NULL	14	Drama
14	$ 2500 Bride, The	1912	NULL	14	Romance

`k-way` Joining

3 Tables

Command

```
> SELECT *
-> FROM movies AS m
-> JOIN movies_genres AS g
-> ON m.id = g.movie_id
-> JOIN directors_genres AS d
-> ON g.genre = d.genre
-> LIMIT 20;
```

Output

id	name	year	rankscore	movie_id	genre	director_id	genre	prob
5266	Abuelitos	1999	6	5266	Short	2	Short	1
5261	Abuela's Revolt	2001	NULL	5261	Short	2	Short	1
...	...	...	...	...	...	...	...	...
5180	Absentminded	1925	NULL	5180	Short	2	Short	1
5178	Absent-Minded Waiter, The	1977	7.7	5178	Short	2	Short	1

3 Tables

Command

```
> SELECT m.id, d.director_id, m.year, m.name, m.rankscore, d.prob, g.genre
-> FROM movies AS m
-> JOIN movies_genres AS g
-> ON m.id = g.movie_id
-> JOIN directors_genres AS d
-> ON g.genre = d.genre
-> LIMIT 20;
```

Output

id	director_id	year	name	rankscore	prob	genre
5266	2	1999	Abuelitos	6	1	Short
5261	2	2001	Abuela's Revolt	NULL	1	Short
...	...	...	...	...	...	...
5180	2	1925	Absentminded	NULL	1	Short
5178	2	1977	Absent-Minded Waiter, The	7.7	1	Short

4 Tables

Command

```
> SELECT m.id, m.year, dn.first_name, dn.last_name, m.name, m.rankscore, d.prob, g.genre
-> FROM movies AS m
-> JOIN movies_genres AS g ON m.id = g.movie_id
-> JOIN directors_genres AS d ON g.genre = d.genre
-> JOIN directors AS dn ON m.id = dn.id
-> LIMIT 20;
```

Output

id	year	first_name	last_name	name	rankscore	prob	genre
3418	1988	Adam	Auslander	A la altura de los ojos	NULL	1	Short
3417	1952	Carlo	Ausino	A kpzett beteg	NULL	1	Short
...	...	...	...	...	...	...	...
3356	1975	Cécile	Aubry	A contra vuelta	NULL	1	Short
3355	1995	Jimmy	Aubrey	A comme acteur	NULL	1	Short

LEFT OUTER JOIN or LEFT JOIN

- `JOINs` the tables considering `LEFT` table as the base table.
- In the `RIGHT` table, if it doesn't find a value - simply takes it as `NULL`.

Note -

- Same concept is applied for `RIGHT JOIN` - keeping `RIGHT` table as the base table.
- For `FULL JOIN` - makes sure all the values are taken care.

Command

```
> SELECT m.name, g.genre
-> FROM movies as m
-> LEFT JOIN movies_genres as g
-> ON m.id = g.movie_id
-> LIMIT 20;
```

Output

name	genre
173 St.G.B. Blutschande	NULL
51 StGB	NULL
...	...
bientt, j'espre	Documentary
Biribi, disciplinaires franais	NULL

Questions

1. Display all the actors acted in the movie which is like 'Green Mile%'
2. Display all the movies acted by particular actor ('Tom Hanks')

Ans 1)

Command

```
> SELECT a.first_name, a.last_name, r.role
-> FROM actors AS a
-> JOIN roles AS r
-> ON a.id = r.actor_id
-> JOIN movies AS m
-> ON m.id = r.movie_id
-> WHERE m.name LIKE 'Green Mile%';
```

Output

first_name	last_name	role
Brent	Briscoe	Bill Dodge
David E.	Browning	Reverend at Funeral
...	...	...
Rachel	Singer	Cynthia Hammersmith
Edrie	Warner	Lady in Nursing Home

Ans 2)

Command

```
> SELECT m.year, m.name, r.role, m.rankscore
-> FROM actors AS a
-> JOIN roles AS r
-> ON r.actor_id = a.id
-> JOIN movies AS m
-> ON m.id = r.movie_id
-> WHERE a.first_name = 'Tom' AND a.last_name = 'Hanks';
```

Output

year	name	role	rankscore
1989	'burbs, The	Ray Peterson	5.9
2003	'Catch Me If You Can': Behind the Camera	Himself	NULL
...	...	...	...
1997	"Celebrity Profile"	Himself	NULL
1998	"From the Earth to the Moon"	Jean-Luc Despont/Host/Narrato	NULL

Command

```
> SELECT m.year, m.name, r.role, m.rankscore
-> FROM actors AS a
-> JOIN roles AS r
-> ON r.actor_id = a.id
-> JOIN movies AS m
-> ON m.id = r.movie_id
-> WHERE a.first_name = 'Tom' AND a.last_name = 'Hanks' AND m.rankscore IS NOT NULL
-> ORDER BY m.year;
```

Output

year	name	role	rankscore
1980	He Knows You're Alone	Elliot	3.7
1984	Splash	Allen Bauer	6.3
...	...	...	...
2004	Ladykillers, The	Professor G.H. Dorr	6.5
2004	Terminal, The	Viktor Navorski	7.1