

# SQL: Queries on One Table

---

- Queries
- SQL Query Language
- Problem-solving in SQL
- Views
- Exercise: Queries on Beer Database

## ❖ Queries

A **query** is a **declarative program** that retrieves data from a database.

declarative = say what we want, not method to get it

Queries are used in two ways in RDBMSs:

- **interactively** (e.g. in `psql`)
  - the entire result is displayed in tabular format on the output
- **by a program** (e.g. in a PLpgSQL function)
  - the result tuples are consumed one-at-a-time by the program

SQL is based on the **relational algebra**, which we discuss elsewhere

## ❖ SQL Query Language

---

An SQL **query** consists of a sequence of clauses:

```
SELECT    projectionList
FROM      relations/joins
WHERE     condition
GROUP BY  groupingAttributes
HAVING    groupCondition
```

FROM, WHERE, GROUP BY, HAVING clauses are optional.

Result of query: a relation, typically displayed as a table.

Result could be just one tuple with one attribute (i.e. one value) or even empty

## ❖ SQL Query Language (cont)

---

Functionality provided by SQL ...

**Filtering:** extract attributes from tuples, extract tuples from tables

```
SELECT b, c FROM R(a, b, c, d) WHERE a > 5
```

**Combining:** merging related tuples from different tables

```
... FROM R(x, y, z) JOIN S(a, b, c) ON R.y = S.a
```

**Summarising:** aggregating values in a single column

```
SELECT avg(mark) FROM ...
```

**Set operations:** union, intersection, difference

## ❖ SQL Query Language (cont)

More functionality provided by SQL ...

**Grouping:** forming subsets of tuples sharing some property

```
... GROUP BY R. a
```

(forms groups of tuples from R sharing the same value of a)

**Group Filtering:** selecting only groups satisfying a condition

```
... GROUP BY R. a HAVING max(R. a) < 75
```

**Renaming:** assign a name to a component of a query

```
SELECT a as name  
FROM Employee(a, b, c) e WHERE e. b > 50000
```

## ❖ SQL Query Language (cont)

Schema:

- *Students(id, name, ...)*
- *Enrolments(student, course, mark, grade)*

Example SQL query on this schema:

```
SELECT    s.id, s.name, avg(e.mark) as avgMark
FROM      Students s
          JOIN Enrolments e on (s.id = e.student)
GROUP BY  s.id, s.name
-- or --
SELECT    s.id, s.name, avg(e.mark) as avgMark
FROM      Students s, Enrolments e
WHERE     s.id = e.student
GROUP BY  s.id, s.name
```

## ❖ SQL Query Language (cont)

---

How the example query is computed:

- produce all pairs of *Students*, *Enrolments* tuples which satisfy condition (*Students.id* = *Enrolments.student*)
- each tuple has (*id*, *name*, ..., *student*, *course*, *mark*, *grade*)
- form groups of tuples with same (*id*, *name*) values
- for each group, compute average mark
- form result tuples (*id*, *name*, *avgMark*)

## ❖ Problem-solving in SQL

---

Starts with an information request:

- (informal) description of the information required from the database

Ends with:

- a list of tuples that meet the requirements in the request

Pre-req: [know your schema](#)

Look for keywords in request to identify required data :

- tell me the [names](#) of all [students](#)...
- [how many](#) [students](#) failed ...
- what is the [highest mark](#) in ...
- which [courses](#) are ... (course codes?)



## ❖ Problem-solving in SQL (cont)

---

Developing SQL queries ...

- relate required data to **attributes** in schema
- identify which **tables** contain these attributes
- combine data from relevant tables (**FROM, JOIN**)
- specify conditions to select relevant data (**WHERE**)
- [optional] define grouping attributes (**GROUP BY**)
- develop expressions to compute output values (**SELECT**)

## ❖ Problem-solving in SQL (cont)

---

**Example:** just the beers that John likes

- which table contains info about beers that are liked?
- Likes (drinker, beers)
- only want tuples where drinker is John (WHERE)
- only want beer names (SELECT beer)

... giving ...

```
select beer from Likes where drinker='John' ;
```

## ❖ Views

A **view** associates a name with a query:

- `CREATE VIEW viewName [ ( attributes ) ] AS Query`

Each time the view is invoked (in a `FROM` clause):

- the *Query* is evaluated, yielding a set of tuples
- the set of tuples is used as the value of the view

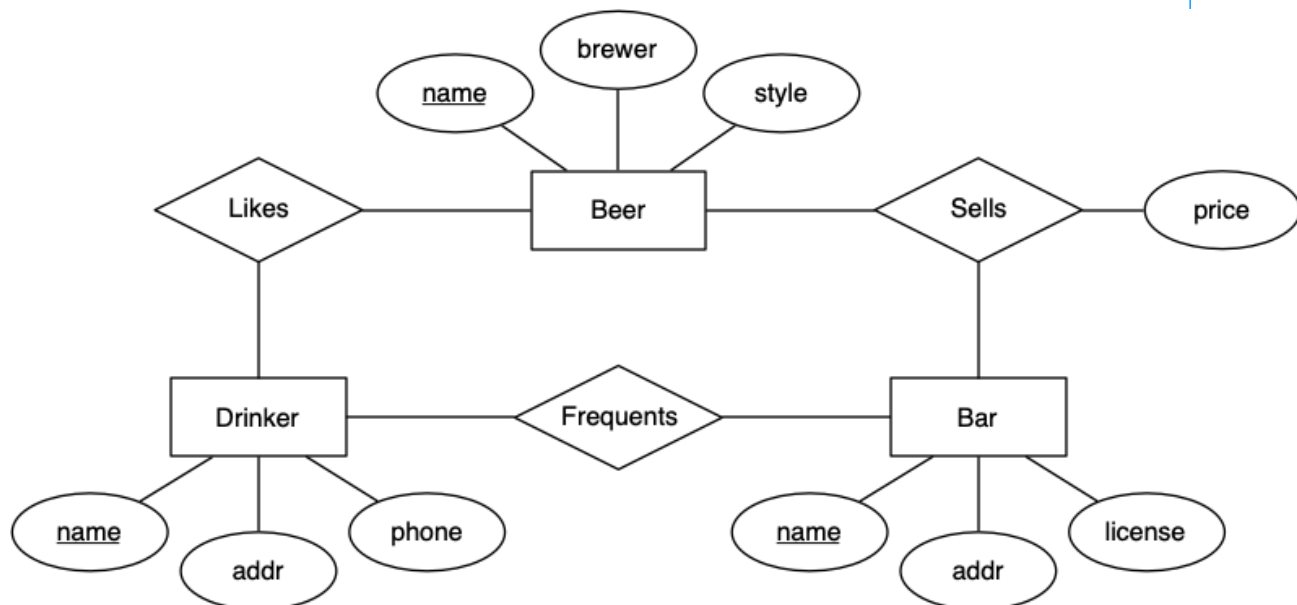
A view can be treated as a "virtual table".

Views are useful for "packaging" a complex query to use in other queries.

cf. writing functions to package computations in programs

## ❖ Exercise: Queries on Beer Database

ER design for Beer database:



COMP3311 21T1 ♦ SQL: Queries on One Table ♦ [11/12]

## ❖ Exercise: Queries on Beer Database (cont)

Answer these queries on the Beer database:

1. What beers are made by Toohey's?
2. Show beers with headings "Beer", "Brewer".
3. How many different beers are there?
4. How many different brewers are there?
5. Which beers does John like?
6. Find pairs of beers by the same manufacturer.
7. How many beers does each brewer make?
8. Which brewers make only one beer?
9. Which brewer makes the most beers?

Produced: 27 Feb 2021