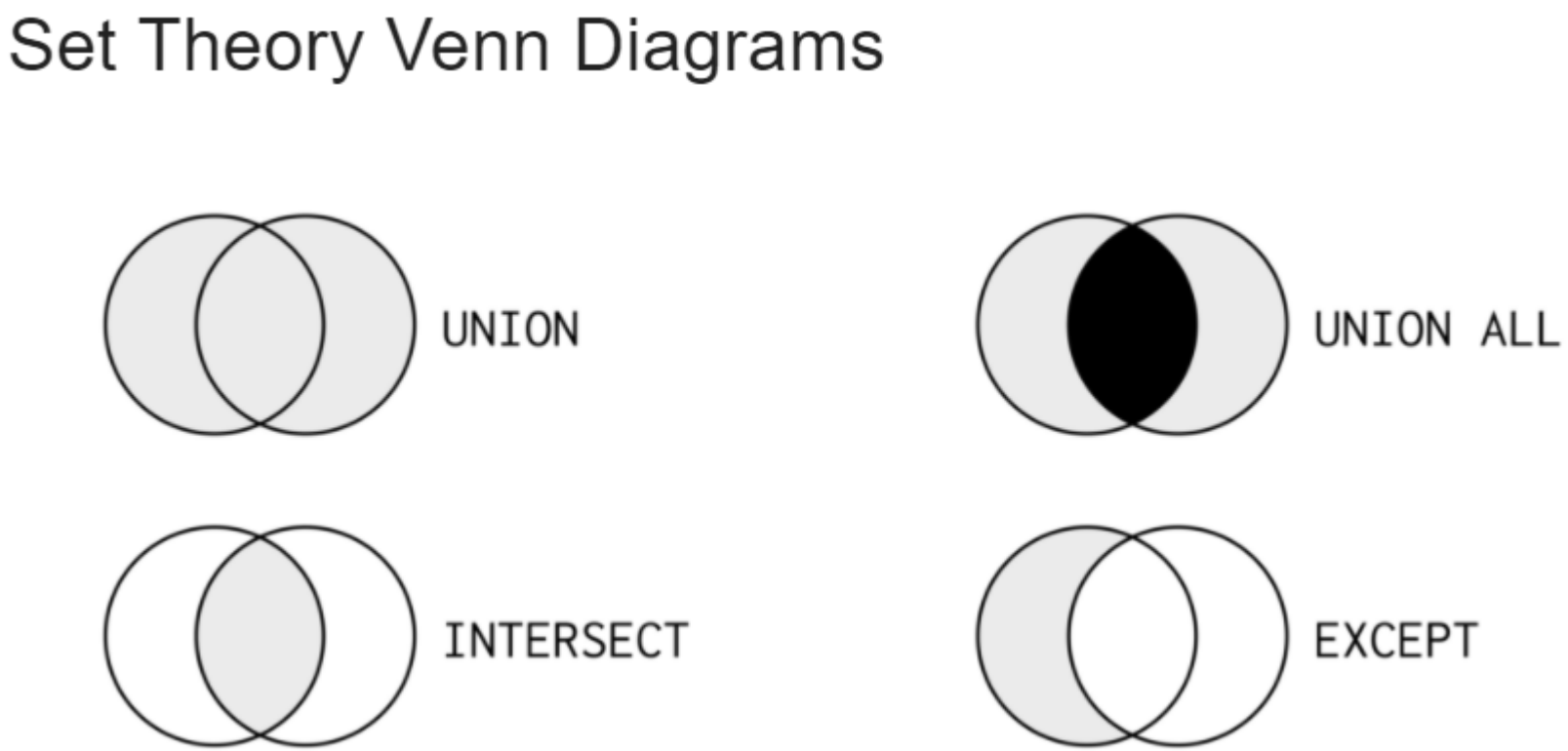
Set Clauses: operations on rows

- Union
- Union all
- Intersect
- Except

The union all and union are different, then later double counting the common values of the two sets.



Joins: operations on columns

- Inner Join

    - An inner join will return only records where the join keys exist within the intersection of the joined tables. The keys are in that intersection. Hence, it is called inner joins. An outer join will return all the records in the intersection, as well as the records outside the intersection. That is why called outer joins.
    - A special case is self-join (No built-in SQL syntax). Using alias to treat as different tables.
    - Intuitive example: show me all students with lockers.
- Outer Join
    - Left join example: show me all students, with their corresponding locker if they have one
    - Right join example: show me all lockers, and the students assigned to them if there are any
    - Full join example: show me all students and all lockers, and match them up where you can
- Cross Join
    - Create all possible cases among two joining tables.
- Semi/Anti Join
    - IN, NOT IN, EXITS etc. are used. No built-in SQL syntax. Semi-join and anti-join should belong to the subquery category.

Joins vs Sub-query (nested query)

- Among all the table operations, set clauses and cross joins are either not so commonly used or straightforward. All other actions are generally into two groups: Joins (three outer joins plus inner join) and Sub-query.
- Performance vs logical clarity. Usually sub-queries are logically clearer. However, explicit joins usually win in performance. As optimizers get better and better, the subquery can be with high performance. Use EXPLAIN to see how your database executes the query on your data. PostgreSQL can rewrite a subquery to a join or a join to a subquery when it thinks one is faster than the other.

General Routing Maps

- A clear understanding of the querying logical process order is essential in writing correct SQL queries. Always resort to the processing order when a question occurs in query.
- A clear understanding of the tool kit for queries involving table relations.
    - On Rows
        - Set clauses: union, union all, except, intersect
    - On Columns
        - Joins: Inner join (self, non-self), outer join (left, right, full).
        - Subquery: these also include semi/anti joins.

Table joining examples

```
SELECT c.name AS country, continent, l.name AS language, official
FROM countries AS c
INNER JOIN languages AS l
USING (code)
```

Note the bracket for code.

Inner Join with itself to achieve some purpose: calculate, e.g. percentage increase.

```
SELECT p1.country_code,
       p1.size AS size2010,
       p2.size AS size2015,
       ((p2.size - p1.size)/p1.size * 100.0) AS growth_perc
FROM populations AS p1
INNER JOIN populations AS p2
ON p1.country_code = p2.country_code
AND p1.year = p2.year - 5;
```

The last condition is added because we are calculating the percentage increase between 2010 to 2015. Note without the last sentence, for each country, there will be four rows.

If-then structure in join structure.

```
select name, continent, code, surface_area,
       case when surface_area > 2000000
             Then 'large'
       when surface_area > 350000
             Then 'medium'
       else 'small' end
       AS geosize_group
From countries;
```

• A final problem on inner join SELECT country_code, size, CASE WHEN size > 50000000 THEN 'large' WHEN size > 1000000 THEN 'medium' ELSE 'small' END AS popsize_group INTO pop_plus
FROM populations WHERE year = 2015;

SELECT name, continent, geosize_group, popsize_group FROM countries_plus AS c INNER JOIN pop_plus AS p ON c.code = p.country_code ORDER BY geosize_group;

Outer joins and cross joins SELECT c.name AS country, region, l.name AS language, basic_unit, frac_unit FROM countries AS c FULL JOIN languages AS l USING (code) FULL JOIN currencies AS cu USING (code) WHERE c.region LIKE 'M%esia';

Join and subquery equivalence

The following query is focused on sub-queries SELECT name, continent, inflation_rate
FROM countries
INNER JOIN economies
ON countries.code = economies.code
WHERE year = 2015
AND inflation_rate IN (
SELECT MAX(inflation_rate) AS max_inf
FROM (
SELECT name, continent, inflation_rate
FROM countries
INNER JOIN economies
ON countries.code = economies.code
WHERE year = 2015) AS subquery
GROUP BY continent);
The following query solves the same problem but focuses on joins SELECT t1.name AS name, t1.continent AS continent, t2.MaxInflation AS inflation_rate
FROM
(
SELECT continent, name, inflation_rate
FROM countries
INNER JOIN economies
USING (code)
WHERE year = 2015
) as t1
INNER JOIN
(
SELECT continent, max(inflation_rate) AS MaxInflation
FROM countries

```
INNER JOIN economies
USING (code)
WHERE year = 2015
GROUP BY continent
) as t2
ON (t1.continent = t2.continent AND t1.inflation_rate = t2.MaxInflation)

SELECT local_name, subquery.lang_num
FROM countries,
(SELECT code, COUNT(*) AS lang_num
FROM languages
GROUP BY code) AS subquery
WHERE countries.code = subquery.code
ORDER BY lang_num DESC;
```