IT SPECIALIST: DATABASE

DATA ANALYST - GROUP 4E

OUR TEAM

ALDI FAUZAN
UNIVERSITAS TELKOM

MUHAMMAD IKHSAN ANUGRAH
UNIVERSITAS DIAN NUSWANTORO

ALFAN NURROHIM
UNIVERSITAS NAHDLATUL ULAMA YOGYAKARTA

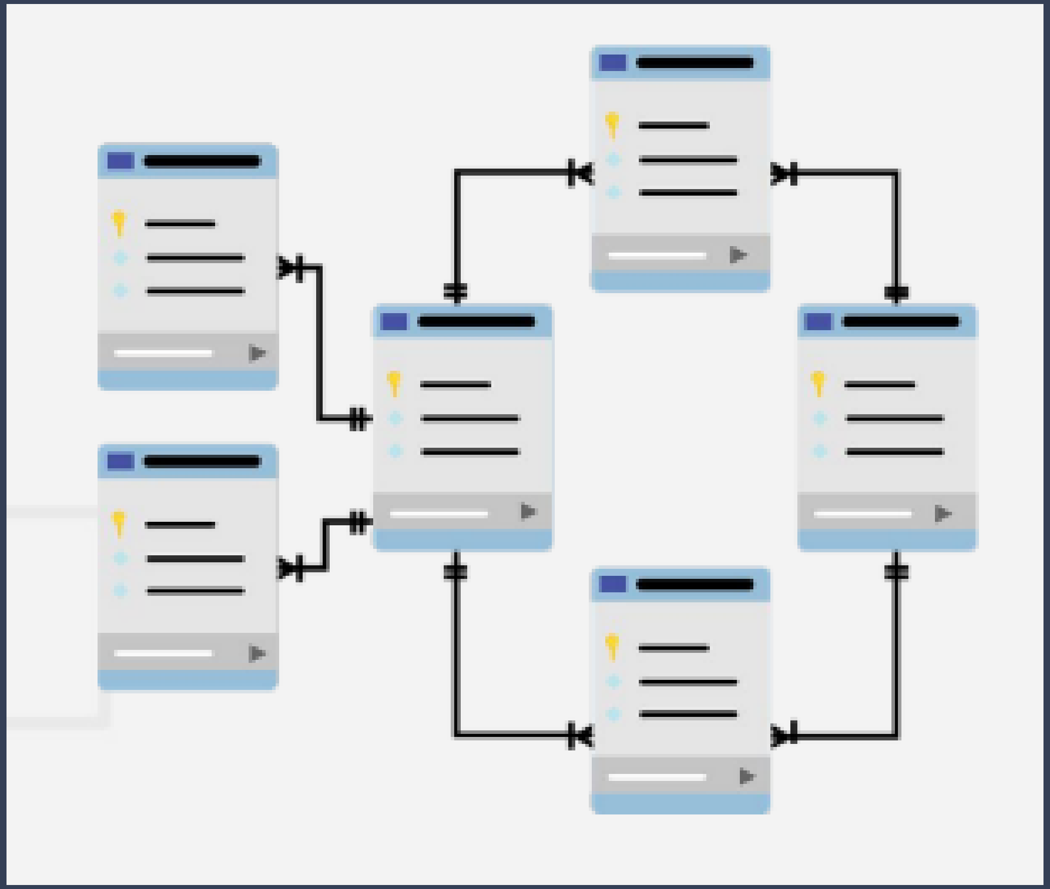
TIA HAERUNISA
UNIVERSITAS SEBELAS APRIL SUMEDANG

FIONA ASTRIDA FUSHIGI
INSTITUT PENDIDIKAN DAN BAHASA CIREBON

WIN AINI TISYA
UNIVERSITAS SULTAN AGENG TIRTAYASA

TABLE OF CONTENT

RELATIONAL DATABASE DESIGN



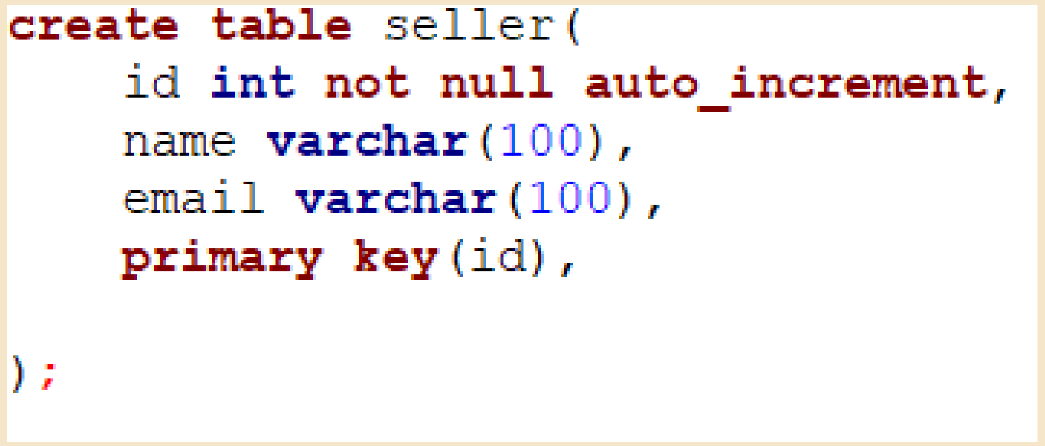
A relational database is a collection of information that organizes data in predefined relationships where data is stored in one or more tables (or "relations") of columns and rows, making it easy to see and understand how different data structures relate to each other.

DATABASE KEYS

Keys in the database are actually ordinary attributes that are simply added to certain query declarations

PRIMARY KEY

The primary key is a unique value that can be used as an identifier for a table, in other words, it is the identity for each row of data in that table.

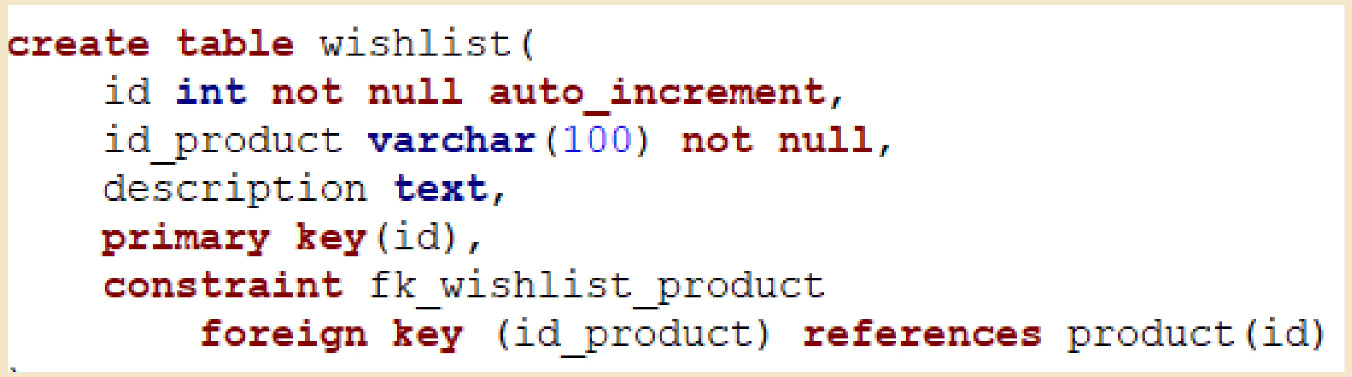


```sql
create table seller(
    id int not null auto_increment,
    name varchar(100),
    email varchar(100),
    primary key(id),

);
```

FOREIGN KEY

A Foreign Key is a marker used to link one row in one table to another row in a different table.

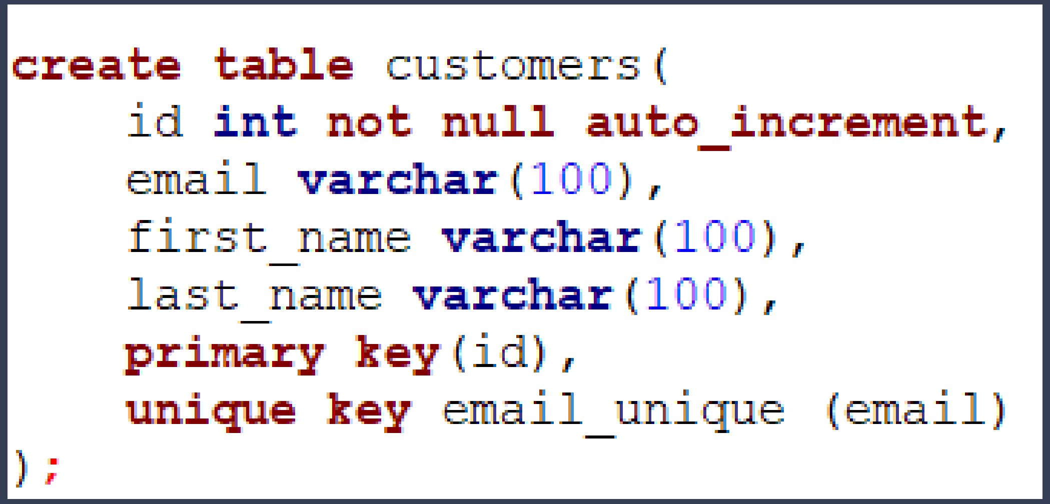


```sql
create table wishlist(
    id int not null auto_increment,
    id_product varchar(100) not null,
    description text,
    primary key(id),
    constraint fk_wishlist_product
        foreign key (id_product) references product(id)
```

DATABASE KEYS

UNIQUE KEY

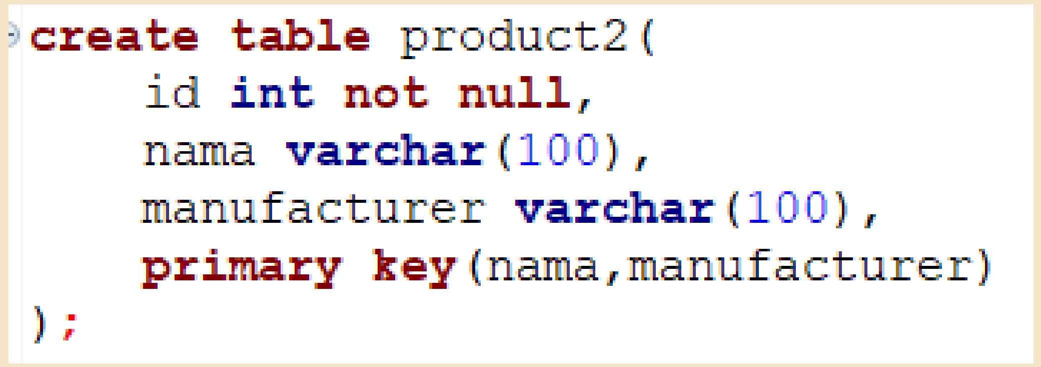
A Unique Key is a unique identifier within a single table.



```sql
create table customers(
    id int not null auto_increment,
    email varchar(100),
    first_name varchar(100),
    last_name varchar(100),
    primary key(id),
    unique key email_unique (email)
);
```

COMPOSITE

A Composite Key is a key that consists of two or more unique attributes.



```sql
create table product2(
    id int not null,
    nama varchar(100),
    manufacturer varchar(100),
    primary key(nama,manufacturer)
);
```

TABLE RELATIONSHIP

Relationships are meaningful associations between tables that contain related information — they're what make databases useful. Without some connection between tables in a database system.

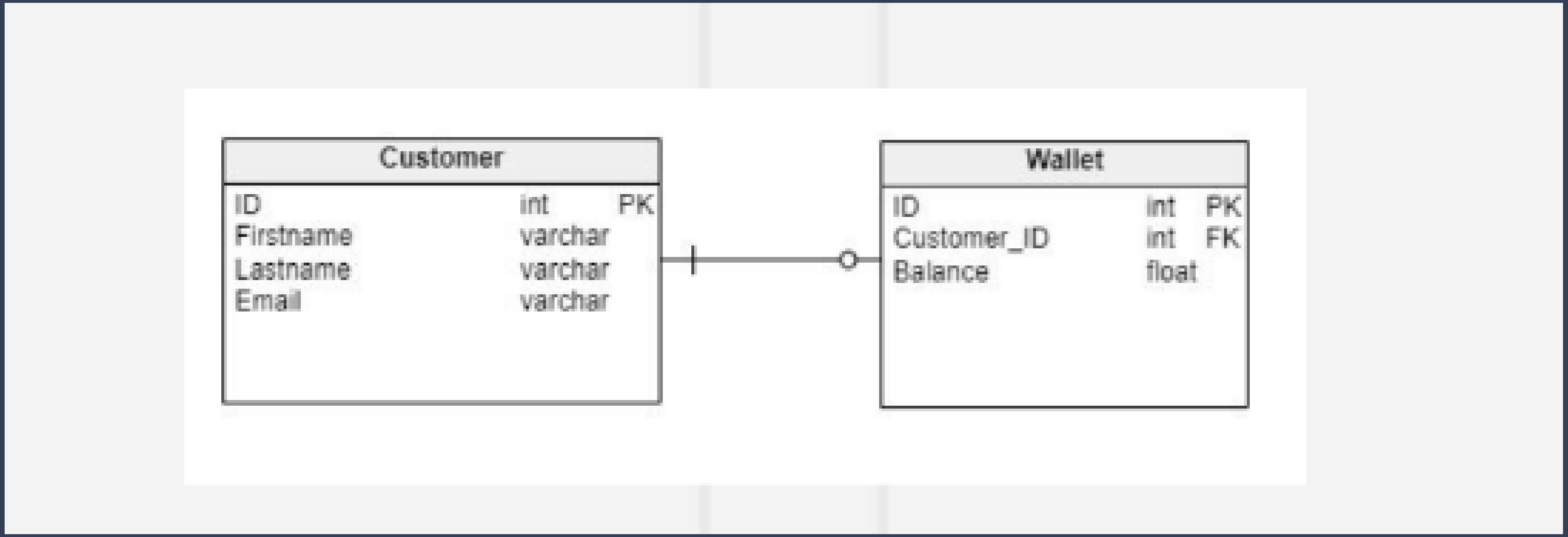


TABLE RELATIONSHIP

ONE-TO-ONE

In a one-to-one relationship, a record in one table can correspond to only one record in another table (or in some cases, no records).

ONE-TO-MANY

One-to-many relationships are the most common type of relationships between tables in a database. In a one-to-many (sometimes called many-to-one) relationship

MANY-TO-MANY

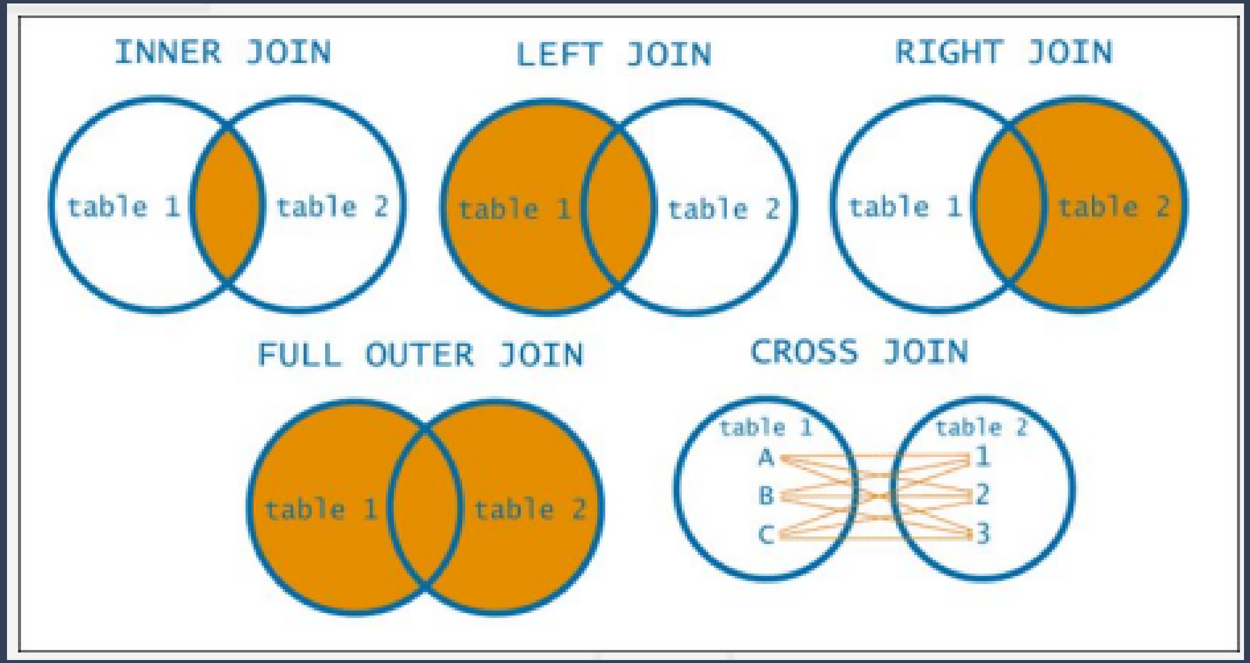
A many-to-many relationship links multiple records in one table to multiple records in another table, typically with each record associated with one or none at all.

JOIN & SET OPERATOR

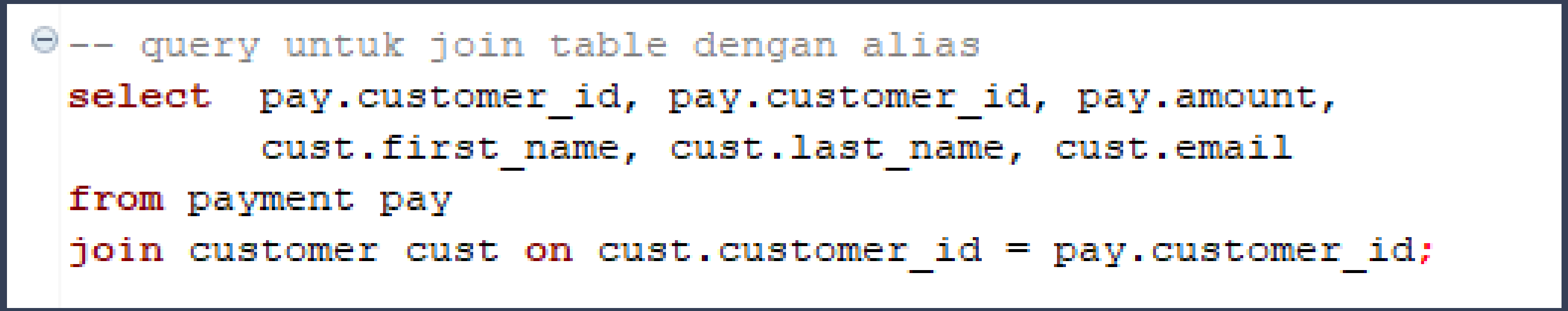
"JOIN" is a simple operation to combine two or more tables into one temporary table, which can then be used for various querying operations.

To perform a join, we need to specify which table serves as a reference to another table. Things to consider when joining:
1. We can directly identify foreign keys (although there are no strict rules stating that you must join using them).
2. The more joins you have, the heavier and slower the query process can become.
3. It's generally advisable not to join more than 5 tables, as it can impact query performance.

Some types of joins:

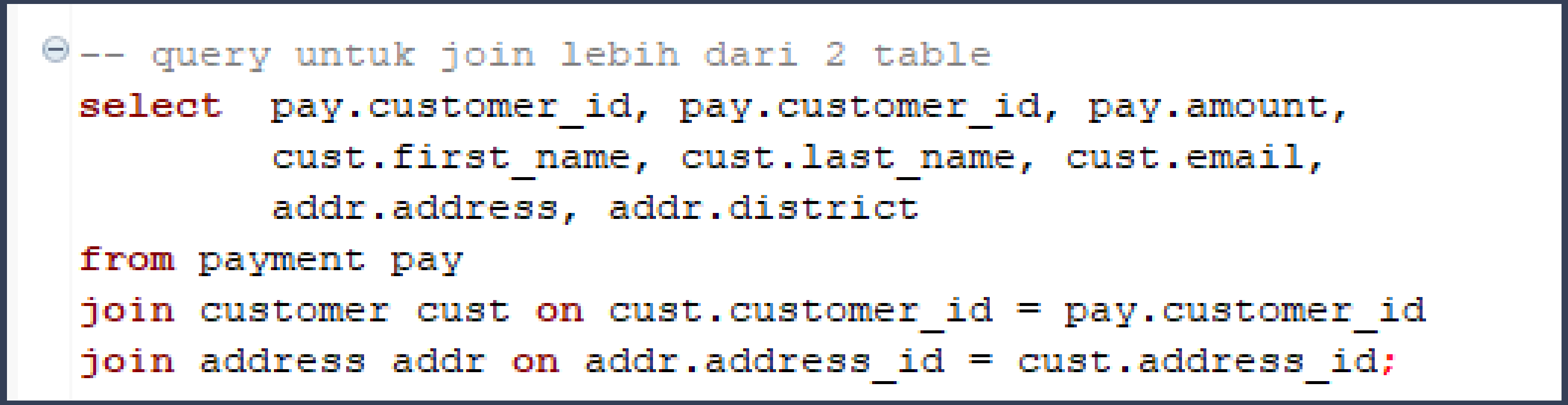


Joining to one table:



```sql
-- query untuk join table dengan alias
select  pay.customer_id, pay.customer_id, pay.amount,
        cust.first_name, cust.last_name, cust.email
from payment pay
join customer cust on cust.customer_id = pay.customer_id;
```

Joining to more than one table:



```sql
-- query untuk join lebih dari 2 table
select  pay.customer_id, pay.customer_id, pay.amount,
        cust.first_name, cust.last_name, cust.email,
        addr.address, addr.district
from payment pay
join customer cust on cust.customer_id = pay.customer_id
join address addr on addr.address_id = cust.address_id;
```

SOME TYPES OF JOIN

INNER JOIN is the default JOIN mechanism in MySQL, where if there is a relationship between the first table and the second table, the result will be displayed. If there is data in the first table that does not have a relationship with the second table or vice versa, the result will not be displayed.

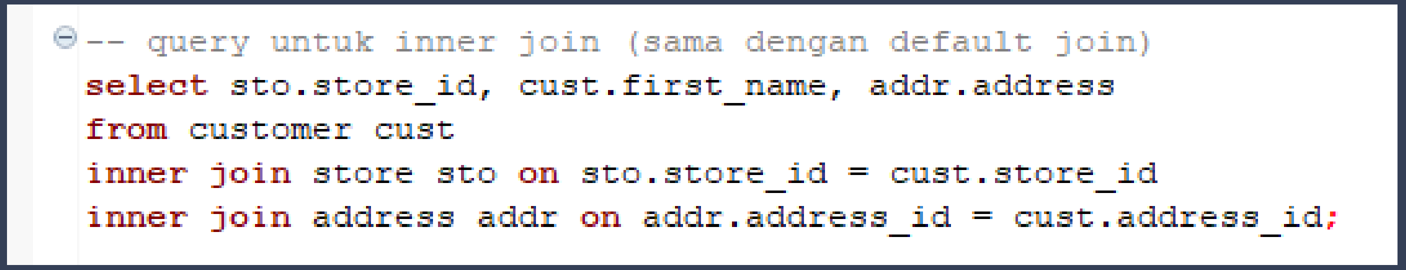
LEFT JOIN is a join mechanism similar to INNER JOIN, but the difference is that all data from the first table will be retrieved, and if there is data that does not have a relationship with the second table, the result will be NULL. In practice, LEFT JOIN is often used more frequently than other types of joins.

RIGHT JOIN is the opposite of LEFT JOIN. The difference is that all data from the second table will be retrieved, and if there is data that does not have a relationship with the first table, the result will be NULL.

CROSS JOIN is one of the joins that is rarely used. In its mechanism, a cross join will multiply the data in the first table with the data in the second table. For example, if there are 5 data in the first table and 5 data in the second table, it will result in 25 data combinations (5 x 5 = 25).

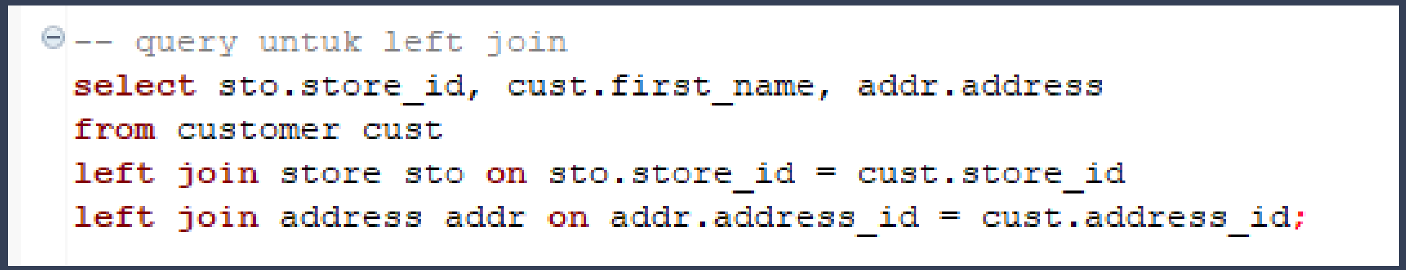
EXAMPLE

INNER JOIN :



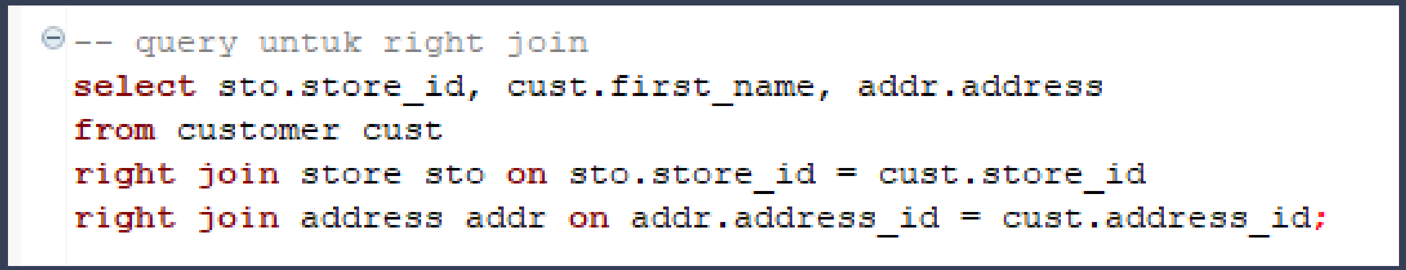
```
-- query untuk inner join (sama dengan default join)
select sto.store_id, cust.first_name, addr.address
from customer cust
inner join store sto on sto.store_id = cust.store_id
inner join address addr on addr.address_id = cust.address_id;
```

LEFT JOIN :



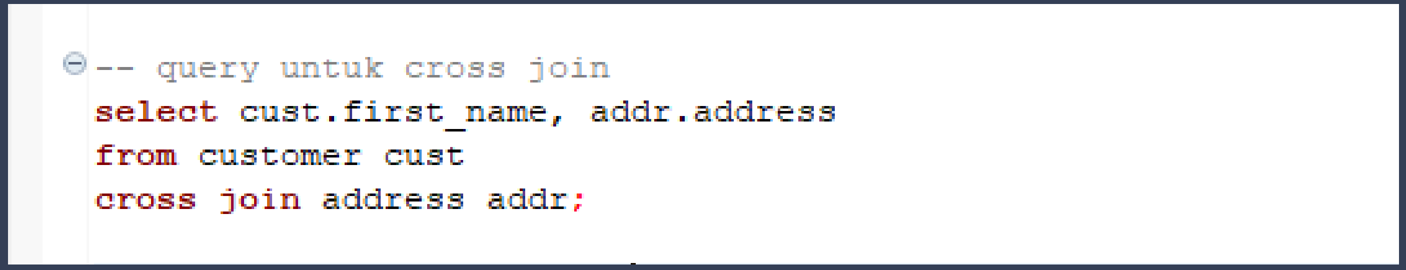
```
-- query untuk left join
select sto.store_id, cust.first_name, addr.address
from customer cust
left join store sto on sto.store_id = cust.store_id
left join address addr on addr.address_id = cust.address_id;
```

RIGHT JOIN :



```
-- query untuk right join
select sto.store_id, cust.first_name, addr.address
from customer cust
right join store sto on sto.store_id = cust.store_id
right join address addr on addr.address_id = cust.address_id;
```

CROSS JOIN :



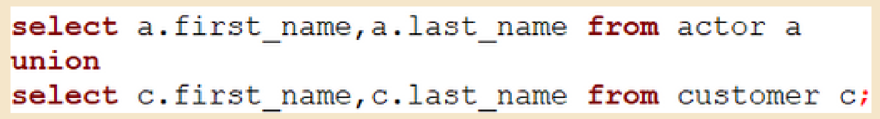
```
-- query untuk cross join
select cust.first_name, addr.address
from customer cust
cross join address addr;
```

SET OPERATOR

Set operators are conditions in which you perform operations using sets generated from two or more different queries. In addition to using the method of joining tables, combining two or more tables can also be achieved using Set Operations.

UNION

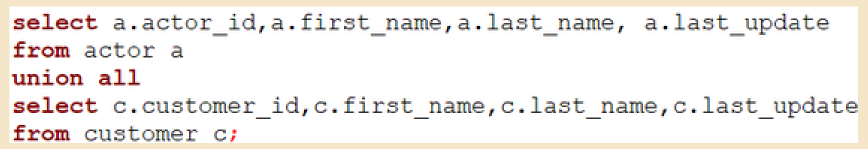
Merges tables based on ID side by side vertically, doesn’t accept duplicate data.



```sql
select a.first_name,a.last_name from actor a
union
select c.first_name,c.last_name from customer c;
```

UNION ALL

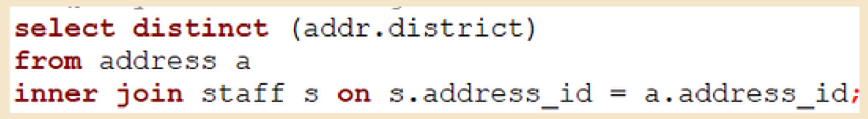
Almost the same as UNION, but in UNION ALL duplicate data will still be displayed in the query results.



```sql
select a.actor_id,a.first_name,a.last_name, a.last_update
from actor a
union all
select c.customer_id,c.first_name,c.last_name,c.last_update
from customer c;
```

INTERSECT

Combining two queries, but only the data contained in the results of the first query and the second query is retrieved.



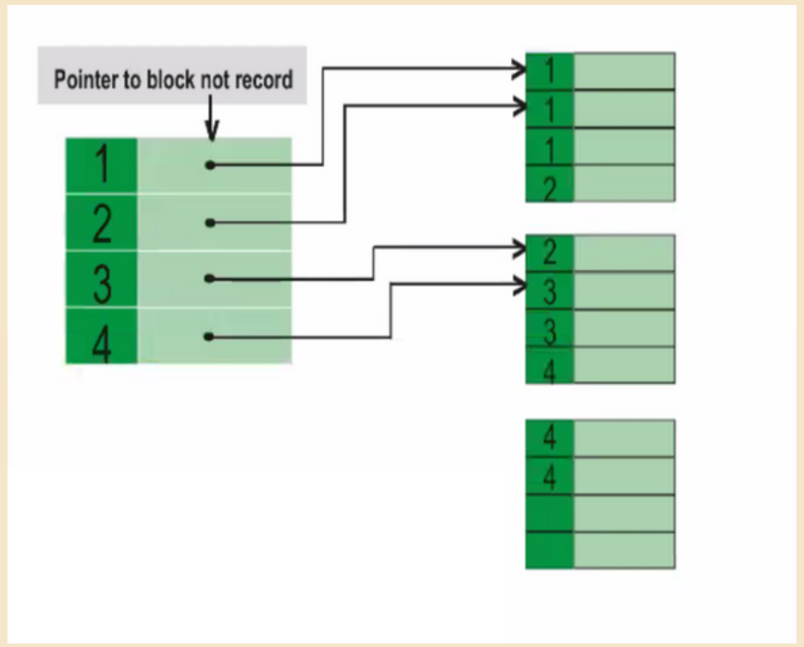
```sql
select distinct (addr.district)
from address a
inner join staff s on s.address_id = a.address_id;
```

INDEX

An object in the database system that can speed up the data query search process.

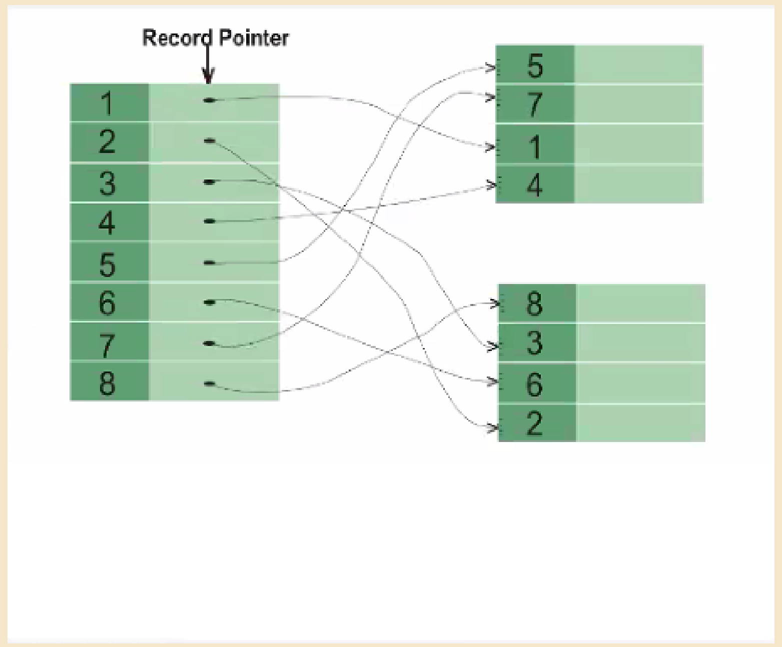
CLUSTERED INDEX

The type of index that rearranges the path of record storage in the table, each data table has only one clustered index.



NON-CLUSTERED INDEX

A type of index that works by providing a reference to the data in a table. Each table can have more than one non-clusterred index.



CLUSTERED VS NON CLUSTERED

CLUSTERED INDEX

- Reformatting the rendering of a table.
- Faster than non-clustered
- Requires less memory
- Can store index data on disk
- One table only has one index

NON CLUSTERED INDEX

- Provides a reference to data in a table.
- Slower than clustered
- Requires more memory
- Cannot save data on disk
- One table can have many indexes

USAGE OF INDEX

NEED INDEX

- Large-sized table.
- Columns frequently used in specific conditions (WHERE, JOIN, etc.)
- Columns containing a wide range of values.
- Columns containing many null values.

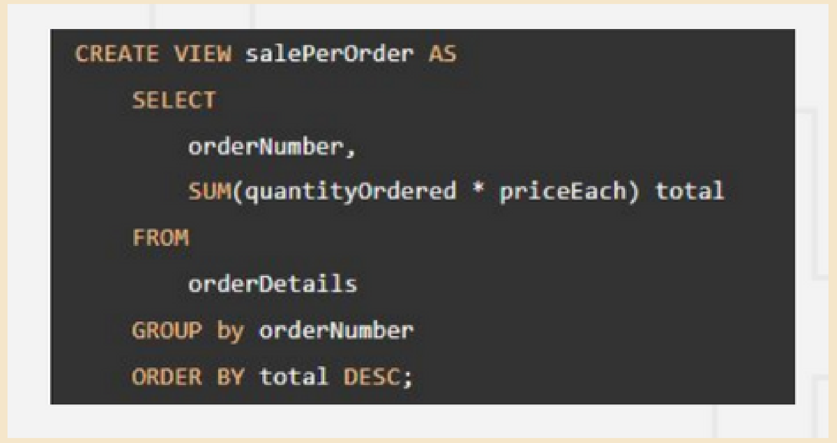
NOT NEEDED INDEX

- Small-sized table
- Columns not frequently used as conditions in queries.
- Columns containing a narrow range of values.
- Table frequently updated.

VIEW

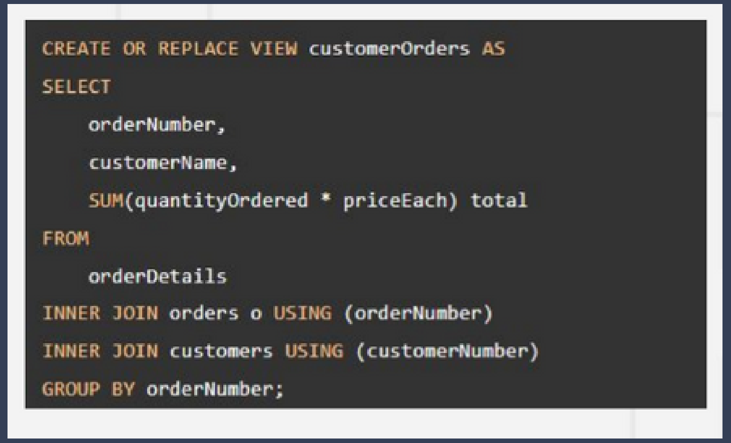
The CREATE VIEW statement creates a new view, or replaces an existing view if the OR REPLACE clause is given. If the view does not exist, CREATE OR REPLACE VIEW is the same as CREATE VIEW. If the view does exist, CREATE OR REPLACE VIEW replaces it.

CREATE VIEW



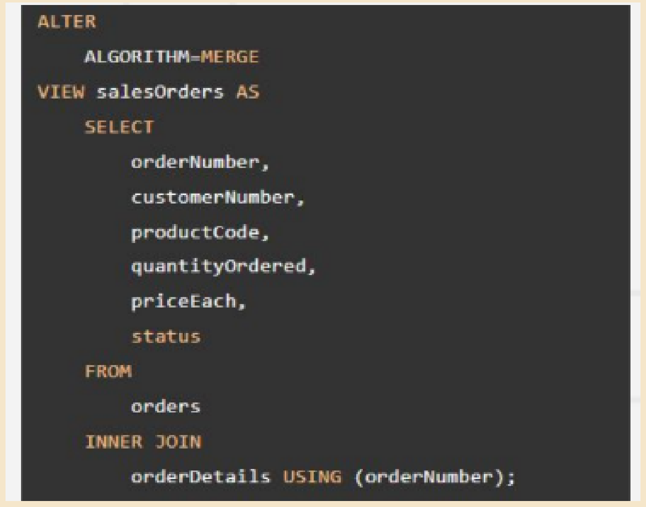
```
CREATE VIEW salePerOrder AS
    SELECT
        orderNumber,
        SUM(quantityOrdered * priceEach) total
    FROM
        orderDetails
    GROUP by orderNumber
    ORDER BY total DESC;
```

USING JOINS



```
CREATE OR REPLACE VIEW customerOrders AS
SELECT
    orderNumber,
    customerName,
    SUM(quantityOrdered * priceEach) total
FROM
    orderDetails
INNER JOIN orders o USING (orderNumber)
INNER JOIN customers USING (customerNumber)
GROUP BY orderNumber;
```

ALTER VIEW



```
ALTER
    ALGORITHM=MERGE
VIEW salesOrders AS
    SELECT
        orderNumber,
        customerNumber,
        productCode,
        quantityOrdered,
        priceEach,
        status
    FROM
        orders
    INNER JOIN
        orderDetails USING (orderNumber);
```

THANK YOU!