IT SPECIALIST: DATABASE

Week 3 - 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

Normalization

The process of organizing data into tables that depict each entity and its relationships. In each step, testing is performed to check for difficulties in inserting, deleting, or updating data. If any issues arise, the relationship is subdivided into more tables. The essence of normalization is:

1. Ensuring strong data integrity.
2. Eliminating redundancy in data.
3. Avoiding anomalies in data.

Normalization Process

The normalization process can be summarized as follows:

- Identify each main entity in the data model.
- Discover the relationships between each entity.
- Determine the attributes possessed by each entity.

Unnormalized Form

1. Each attribute of the relation has only a single balue and there is no repetition of the attribut group in the row
2. Does’nt have repeating attribute groups.

No Fac	Kode Supp	Nama Supp	Kode Brg	Nama Barang	Tanggal	Jatuh Tempo	Qty	Harga	Jumlah	Total
779	S02	Hitachi	R02	Rice Chocker C3	02/02/01	09/03/01	10	150000	1500000	1500000
998	G01	Gobel	A01	AC Split ½ PK	07/02/01	09/03.01	10	135000	13500000	33500000
		Nustra	A02	AC Split 1 PK			10	2000000	20000000	

1st NORMAL FORM

A relation is said to satisfy the first normal form (1NF) if and only if each attribute of the relation has only single values, and there are no repeating groups of attributes within rows.

The goals of forming 1NF are:

1. Making the table's semantics more explicit.
2. Allowing all relational operators to be applied to the table.

No Fac	Kode Supp	Nama Supp	Kode Brg	Nama Barang	Tanggal	Jatuh Tempo	Qty	Harga	Jumlah	Total
779	S02	Hitachi	R02	Rice Chocker C3	02/02/01	09/03/01	10	150000	1500000	1500000
998	G01	Gobel	A01	AC Split ½ PK	07/02/01	09/03.01	10	135000	13500000	33500000
		Nustra	A02	AC Split 1 PK			10	2000000	20000000	

No Fac	Kode Supp	Nama Supp	Kode Brg	Nama Barang	Tanggal	Jatuh Tempo	Qty	Harga	Jumlah	Total
779	S02	Hitachi	R02	Rice Chocker C3	02/02/01	09/03/01	10	150000	1500000	1500000
998	G01	Gobel Nustra	A01	AC Split ½ PK	07/02/01	09/03/01	10	135000	13500000	33500000
998	G01	Gobel Nustra	A02	AC Split 1 PK	07/02/01	09/03/01	10	2000000	20000000	33500000

2nd NORMAL FORM

A relation is said to satisfy the second normal form (2NF) if it meets the 1NF criteria and if every non-primary key attribute is functionally dependent on all the key attributes (which are attributes capable of uniquely identifying each row).

The goals of forming 2NF are:

1. Making the semantics of 2NF tables more explicit.
2. Reducing update anomalies that may still occur in 1NF.

No Fac	Kode Supp	Nama Supp	Kode Brg	Nama Barang	Tanggal	Jatuh Tempo	Qty	Harga	Jumlah	Total
779	S02	Hitachi	R02	Rice Chocker C3	02/02/01	09/03/01	10	150000	1500000	1500000
998	G01	Gobel Nustra	A01	AC Split ½ PK	07/02/01	09/03/01	10	135000	13500000	33500000
998	G01	Gobel Nustra	A02	AC Split 1 PK	07/02/01	09/03/01	10	2000000	20000000	33500000

Relasi Supplier

Kode_Supplier	Nama _Supplier
S02	Hitachi
G01	Gobel Nustra
G01	Gobel Nustra

Relasi Barang

Kode_Barang	Nama_Barang	Harga
R02	Rice Chocker C3	150000
A01	AC Split ½ PK	135000
A02	AC Split 1 PK	2000000

Relasi Faktur

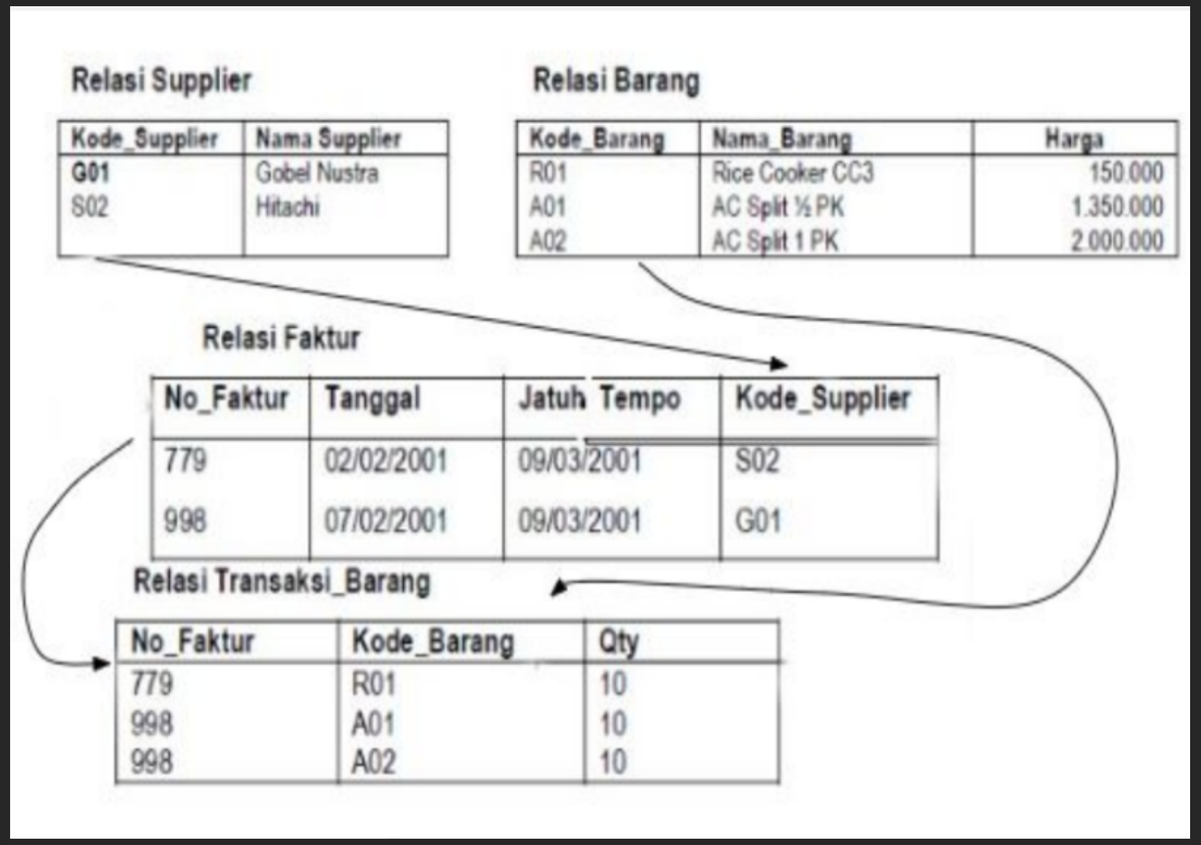
No_Faktur	(Kode_Barang)	(Kode _Supplier)	Tanggal	Jatuh_tempo	Qty
779	R02	S02	02/02/01	09/03/01	10
998	A01	G01	07/02/01	09/03/01	10
998	A02	G01	07/02/01	09/03/01	10

3rd NORMAL FORM

A relation is said to satisfy the third normal form (3NF) if it satisfies 2NF and every non-key attribute is not functionally dependent on any other non-key attribute in the relation (there is no transitive dependency on non-key attributes).

The goals of forming 2NF are:

1. Making the semantics of 3NF tables more explicit (full functional dependency only on the primary key).
2. Preventing update anomalies that may still occur in 2NF.



Relasi Supplier

Kode_Supplier	Nama Supplier
G01	Gobel Nustra
S02	Hitachi

Relasi Barang

Kode_Barang	Nama_Barang	Harga
R01	Rice Cooker CC3	150.000
A01	AC Split ½ PK	1.350.000
A02	AC Split 1 PK	2.000.000

Relasi Faktur

No_Faktur	Tanggal	Jatuh Tempo	Kode_Supplier
779	02/02/2001	09/03/2001	S02
998	07/02/2001	09/03/2001	G01

Relasi Transaksi_Barang

No_Faktur	Kode_Barang	Qty
779	R01	10
998	A01	10
998	A02	10

Subquery

A subquery, also known as a nested query or inner query, is a query within another SQL (Structured Query Language) query. It is a powerful and versatile feature in relational databases that allows you to retrieve data from one or more tables based on the results of another query. Subqueries are often used to filter, sort, or perform calculations on data before it's used in the main query. They can be employed in various parts of an SQL statement, including the SELECT, FROM, WHERE, HAVING, and JOIN clauses.

Subqueries add flexibility to SQL queries and allow you to perform complex operations on your database's data. They are an essential tool for working with relational databases and extracting meaningful insights from your data.

Rules Of Subquery

GROUP BY can be used in a subquery, but ORDER BY is best avoided in subqueries.

Subqueries must be enclosed in parentheses.

Subqueries that return more than one row can only be used with certain value operators, such as the IN operator.

The BETWEEN operator cannot be used with subqueries, but it can be used within subqueries.

A subquery cannot be immediately enclosed in set operators (UNION, UNION ALL).

Example

Subquery after FROM

```
SELECT MAX(price)
FROM (SELECT price
      FROM categories
               INNER JOIN products
                          ON (products.id_category = categories.id)) as cp;
```

Subquery after WHERE

```
SELECT *
FROM products
WHERE price > (SELECT AVG(price) FROM products);
```

Control Flow

Control Flow (or Flow of Control) is a conditional statement used to execute commands that control the flow of a program based on specific conditions while the program is running. Control Flow in SQL has a concept that is somewhat similar but not as complex as programming languages (such as Python, Java, etc.).

Control Flow refers to the ability to control the flow of execution of SQL statements based on specific conditions. Although SQL is not as complex as programming languages like Python, Java, or C++, it still has some fundamental mechanisms for managing the flow of execution.

Type of Control Flow

CASE

Case statements are used for complex conditions. If a condition is evaluated and returns TRUE, then the corresponding CASE statement is executed.

```
select rental_rate as rr,
    (case
       when rental_rate < 1 then 'Murah'
       when rental_rate between 1 and 3 then 'Normal'
       else 'Mahal'
    end) as harga_kategori
from film f;
```

IF

The IF statement will evaluate the given condition, and if the condition is met (TRUE), then the statement will be executed.

```
select f.rental_rate,
        if
        (f.rental_rate < 1,'Murah',
            if
            (f.rental_rate between 1 and 3,'B','C')
            ) as kategori_harga
from film f;
```

IFNULL

The IFNULL statement is used to return a certain value in a column that is NULL. If there is NULL in the condition, then the statement is executed.

```
select f.rental_rate,
       IFNULL(
           select(f.rental_rate < 1,'Murah',
             if
             (f.rental_rate between 1 and 3,'B','C')
           ),
           'Tidak Diketahui'
       ) as kategori_harga
from film f;
```

Common Table Expression

A Common Table Expression (CTE) is a SQL query construct used to simplify JOIN operations in SQL into subqueries and is capable of providing hierarchical queries. CTEs are known for handling hierarchical and recursive queries. In summary, CTEs are defined using the WITH operator, and we can define one or more CTEs in our query.

```sql
with cte_film as(
    select film_id, title, rating, rental_rate
    from film
), cte_film_actor as(
    select actor_id, film_id
    from film_actor
), cte_actor as(
    select actor_id, first_name, last_name
    from actor
)
select  fil.film_id, fil.title, fil.rating,
        act.first_name, act.last_name
from cte_film fil
left join cte_film_actor fa on fa.film_id = fil.film_id
left join cte_actor act on act.actor_id = fa.actor_id;
```

Advantages of CTE

Readability

Makes it easier to read complex queries that have been created. With a hierarchical pattern, the query will be easier to read compared to making lots of subqueries or creating several separate views.

Recursion

Supports creating recursive queries, where a query can call itself. Very useful when we need to work with hierarchical data, structured with certain patterns.

Thank You!

Presented by **Group 4E**

Kampus Merdeka x MyEduSolve