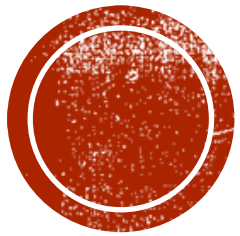


ADVANCED SQL

DigitalSkola

RELATIONAL DATABASE MODEL (THE SCHEMA-BASED CONSTRAINTS)

```
drop table if exists DEPARTMENT3
create table DEPARTMENT3(
id int not null primary key,
department_name varchar (255) not null
);


drop table if exists STUDENT3
create table STUDENT3(
id int not null primary key,
student_name varchar (255) not null,
department_id int not null,
foreign key(department_id)
references DEPARTMENT3(id)
);
```

1. *Create table* dan *drop table* merupakan contoh *command* yang menspesifikasi *integrity constraint.*
2. *Entity integrity* memastikan atribut *primary key* tidak NULL karena menggambarkan entitas pada relasi, jika NULL, tidak bisa bekerja
3. Int, varchar digunakan untuk menspesifikasi jenis data dari atribut yang dipakai.
4. *Foreign key* merupakan *primary key* yakni atribut yang menyatakan hubungan atribut satu dengan yang lain

DigitalSkola

OPERASI PADA *RELATIONAL MODEL*

1. *Insert* digunakan untuk memasukkan data pada relasi
2. *Delete* digunakan untuk menghapus data dari tabel
3. *Modify* digunakan untuk mengubah nilai pada atribut data yang telah ada
4. *Select* digunakan untuk memilih data spesifik

DigitalSkola

MENINGKATKAN PERFORMA *DATABASE*

DigitalSkola

MENGAPA?

1. Semakin besar data yang terdapat pada *database*, maka diperlukan *response time* yang besar pula
2. Data yang disimpan pada *database* adalah tidak berurutan, dan menjadikan proses pencarian tidak efektif

DigitalSkola

CARA MENINGKATKAN PERFORMA *DATABASE*

Indexing Database

Partitioning Database

Where Clause

Join

DigitalSkola

INDEXING DATABASE

Indexing adalah sebuah data struktur yang menyimpan nilai spesifik sebuah kolom pada sebuah tabel. *Indexing* berfungsi Untuk membantu mempercepat proses eksekusi sebuah *query* ke sebuah *database* yang sudah berisi banyak data.

DigitalSkola

CONT...

Contoh:

```
Select * from m_customers Where lastname =  'Dare'
```

Dengan dijalankannya SQL *statement* tersebut, maka akan dilakukan eksplorasi seluruh data di *table* `m_customers` dari data pertama sampai terakhir dan melakukan *filter* dengan `lastname = 'Dare'` di setiap baris. Baris data yang memenuhi persyaratan akan dikumpulkan sebagai hasil.

DigitalSkola

PARTITIONING DATABASE

- Tabel database yang telah dipartisi maka *query* akan *scan* khusus di bagian dimana data itu berada, dan menyebabkan proses eksekusi *query* lebih cepat.
- Manfaat lainnya adalah tabel partisi adalah tiap-tiap *segment* (partisi atau subpartisi) bisa ditaruh di *tablespace* yang berbeda, sehingga user mendapat manfaat dari penyebaran *tablespace*, yaitu penyebaran I/O dan mengurangi resiko *loss data* karena *tablespace corrupt*.

DigitalSkola

JENIS PARTISI

1. *Range Partition*

Pada *range partition*, data dikelompokkan berdasarkan *range* (rentang) nilai yang kita tentukan. *Range partition* ini cocok digunakan pada kolom yang nilainya terdistribusi secara merata. Contoh yang paling sering adalah kolom tanggal.

1. *List Partition*

Pada *list partition*, data dikelompokkan berdasarkan nilainya. Cocok untuk kolom yang variasi nilainya tidak banyak. Disini kita masih menggunakan contoh *table* penjualan. Yang cocok dengan *list partition* adalah kolom area.

DigitalSkola

WHERE CLAUSE

- *Where Clause* merupakan metode *query* data dengan mengambil data sesuai kondisi tertentu, dengan cara ini dapat menyaring *record* sehingga meminimalkan beban jaringan.

Contoh

```
Select * from db_postal_code_area WHERE province_code=11
```

DigitalSkola

JOIN

Join adalah penggabungan *table* yang dilakukan melalui kolom / *key* tertentu yang memiliki nilai terkait untuk mendapatkan satu *set* data dengan informasi lengkap.

Join diperlukan karena perancangan *table* pada sistem transaksional kebanyakan di normalisasi, salah satu alasannya untuk menghindari redundansi data.

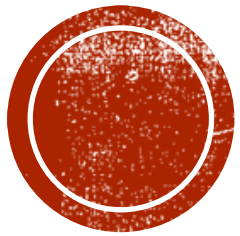
DigitalSkola

CONT...

- Contoh

```
Select * from db_postal_code_data_join db_province_data ON
db_postal_code_data.province_code = db_province_data.province_code;
```

DigitalSkola



ETL *FROM SCRATCH*

DigitalSkola

WEB SCRAPING

- *Web Scraping* adalah proses ekstrasi (E) pada ETL data dari sebuah *website*. Hal ini dapat membantu *user* untuk dapat mengumpulkan data lebih banyak dan terautomatisasi.
- Menggunakan *pandas* untuk membaca table secara spesifik.

DigitalSkola

CONT...

- contoh

```
import pandas as pd
import logging

from typing import List

logging.basicConfig(level=logging.INFO)


def scrape(url: str, header:str = None) -> List[pd.DataFrame]:
    logging.info(f"Scraping website with url: '{url}' ...")
    return pd.read_html(url, header=header)
```

DigitalSkola

CLEANING DATA

- Merupakan proses transformasi data dari seuatu bentuk ke bentuk lain.
- Contohnya merubah milyar menjadi juta untuk sebuah tabel.

DigitalSkola

CONT...

- Contoh

```python
def transform2020(df: pd.DataFrame, tahun: int) -> pd.DataFrame:
    logging.info("Transforming DataFrame ...")

    columns_mapping = {
        "Nomor Urut": "nomor_urut",
        "Nama": "nama",
        "Perusahaan": "perusahaan",
        "Kekayaan Bersih (US$)": "kekayaan_bersih_usd"
    }

    renamed_df = df.rename(columns=columns_mapping)
    renamed_df["tahun"] = tahun
    renamed_df["umur"] = ""
    renamed_df["kekayaan_bersih_usd_juta"] = renamed_df["kekayaan_bersih_usd"].apply(
        lambda value: float(transform_money_format(value)) * 1000 if is_money_miliar(value) else float(transform_money_form
    )

    return renamed_df[["nomor_urut", "tahun", "nama", "perusahaan", "kekayaan_bersih_usd_juta", "umur"]]
```

DigitalSkola

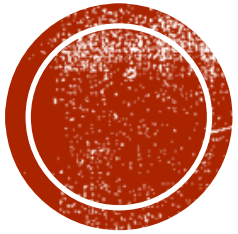
STORING DATA TO DATABASE

- Merupakan proses *Load (L)* dari ETL.

Contoh

```
37  def write_to_postgres_2020(df: pd.DataFrame, db_name: str, table_name: str, connection_string: str) -> None:
38      engine = create_engine(connection_string)
39
40      logging.info(f"Writing dataframe to database: '{db_name}', table: '{table_name}' ...")
41      df.to_sql(table_name, con=engine, if_exists="append", index=False)
```

DigitalSkola

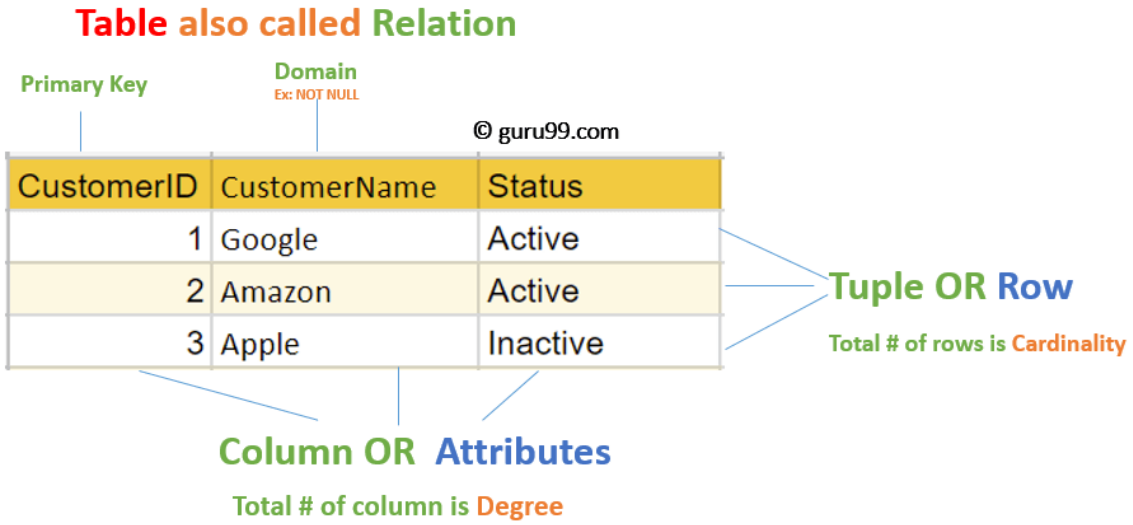


DATA WAREHOUSE & DATA MODELLING

DigitalSkola

RELATIONAL DATA MODEL

- Menggambarkan data, relasi antar data tersebut dan semantiknya
- Berbasis *record* karena memiliki format data yang *fix* (kolom dan baris)
- Setiap relasi harus memiliki *unique value*
- Relasi tidak boleh memiliki 2 kolom/atribut dengan nama yang sama



DigitalSkola

RELATIONAL MODEL CONSTRAINT - ENTITY INTEGRITY

- Setiap baris data didalam sebuah *table* harus memiliki sebuah *key* yang unik dan tidak *null* sehingga baris data tersebut dapat dibedakan dari baris data yang lain menggunakan *Primary Key*

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
```

DigitalSkola

RELATIONAL MODEL CONSTRAINT - REFERENTIAL INTEGRITY

- Aturan terhadap relasi antar tabel untuk menjamin validasi hubungan antara *record* didalam tabel yang terkait menggunakan *Foreign Key*

```
CREATE TABLE Orders (
    OrderID int NOT NULL PRIMARY KEY,
    OrderNumber int NOT NULL,
    PersonID int FOREIGN KEY REFERENCES Persons(PersonID)
);
```

DigitalSkola

RELATIONAL MODEL CONSTRAINT – CHECK CLAUSE

- Memastikan agar ketika terdapat *row* baru yang dimasukkan dalam relasi, maka harus memenuhi kondisi di dalam perintah *check*

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

DigitalSkola

RELATIONAL MODEL CONSTRAINT - NOT NULL CONSTRAINT

- Setiap *field* akan diberikan perintah agar selalu terisi (tidak boleh bernilai *null*)

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
);
```

DigitalSkola

RELATIONAL MODEL CONSTRAINT - DEFAULT VALUE CONSTRAINT

- Digunakan untuk menetapkan nilai *default* pada *field.*

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT 'Sandnes'
);
```

DigitalSkola

thank you!

DigitalSkola