

IF2240 - Basis Data

Tugas Besar

Milestone 3: *Integrity Constraints*



Disusun Oleh:

Hazelnut Milk Tea with Pudding

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

IDENTITAS KELOMPOK

Nomor Kelompok	: 10
Kode Kelompok	:GWS
Nama Kelompok	:Hazelnut Milk Tea with Pudding
Anggota Kelompok	: 1.Dwi Kalam Amal Tauhid (13519210) 2. Akbar Al Fattah (13522036) 3. Imanuel Sebastian Girsang (13522058) 4. Ahmad Mudabbir Arif (13522072) 5. Sa'ad Abdul Hakim (13522092)
Kelas	: K2
Dosen Pengampu	: Dr. Fazat Nur Azizah, S.T, M.Sc
Problem Set	: Apple Music
Nama Asisten	: Anthonio Natthan Krishna (13521162)
Tanggal Pengumpulan	: 18 Mei 2024

1. FUNCTIONAL DEPENDENCIES

Tabel	apple_id
Daftar Atribut	<ul style="list-style-type: none"> • id (PK) • email • nama_pengguna • no_telp_recovery
Daftar Non-Trivial F _c	$\text{id} \rightarrow \text{email nama_pengguna no_telp_recovery}$

Tabel	playlist
Daftar Atribut	<ul style="list-style-type: none"> • playlist_id (PK) • pengguna_id (PK,FK) • nama
Daftar Non-Trivial F _c	$\text{playlist_id pengguna_id} \rightarrow \text{nama}$

Tabel	isi_playlist
Daftar Atribut	<ul style="list-style-type: none"> • playlist_id (PK,FK) • pengguna_id (PK,FK) • lagu_id (PK,FK) • produk_komersial_id (PK,FK)
Daftar Non-Trivial F _c	(semuanya trivial)

Tabel	lagu_produk_komersial
Daftar Atribut	<ul style="list-style-type: none"> • lagu_id (PK,FK) • produk_komersial_id (PK,FK)
Daftar Non-Trivial F _c	(semuanya trivial)

Tabel	produk_komersial
Daftar Atribut	<ul style="list-style-type: none"> • id (PK) • artis_id (FK) • judul • tipe • genre • tanggal_rilis
Daftar Non-Trivial F _c	$\text{id} \rightarrow \text{artis_id judul tipe genre tanggal_rilis}$

Tabel	subscription
Daftar Atribut	<ul style="list-style-type: none"> • subscription_id (PK) • pengguna_id (PK,FK) • subscription_plan_jenis (FK) • tanggal_subscribe • tanggal_expired • status
Daftar Non-Trivial F _c	$\text{subscription_id pengguna_id} \rightarrow \text{subscription_plan_jenis}$ $\text{tanggal_subscribe tanggal_expired status}$

Tabel	lirik
Daftar Atribut	<ul style="list-style-type: none"> • line (PK) • lagu_id (PK,FK) • writer_id (FK) • text
Daftar Non-Trivial F _c	$\text{line lagu_id} \rightarrow \text{writer_id text}$

Tabel	kualitas_audio_lagu
Daftar Atribut	<ul style="list-style-type: none"> • lagu_id (PK,FK) • kualitas_audio (PK)
Daftar Non-Trivial F _c	(semuanya trivial)

Tabel	lagu
Daftar Atribut	<ul style="list-style-type: none"> • id (PK) • artis_id (FK) • label_id (FK) • judul • durasi • tanggal_rilis
Daftar Non-Trivial F _c	$\text{id} \rightarrow \text{artis_id label_id judul durasi tanggal_rilis}$

Tabel	subscription_plan
Daftar Atribut	<ul style="list-style-type: none"> • jenis (PK) • harga_per_bulan
Daftar Non-Trivial F _c	$\text{jenis} \rightarrow \text{harga_per_bulan}$

Tabel	video_musik
Daftar Atribut	<ul style="list-style-type: none"> • id (PK) • artis_id (FK) • lagu_id (FK) • label_id (FK) • judul • durasi • tanggal_rilis
Daftar Non-Trivial F _c	$\text{id} \rightarrow \text{artis_id lagu_id label_id judul durasi tanggal_rilis}$

Tabel	label
Daftar Atribut	<ul style="list-style-type: none"> • id (PK) • nama • tahun_berdiri • asal_negara
Daftar Non-Trivial F _c	$\text{id} \rightarrow \text{nama tahun_berdiri asal_negara}$

Tabel	video_extra
Daftar Atribut	<ul style="list-style-type: none"> • id (PK) • artis_id (FK) • label_id (FK) • judul • durasi • tanggal_rilis
Daftar Non-Trivial F _c	$\text{id} \rightarrow \text{artis_id}$ label_id judul durasi tanggal_rilis

Tabel	host_video_extra
Daftar Atribut	<ul style="list-style-type: none"> • host_id (PK,FK) • video_extra_id(PK,FK)
Daftar Non-Trivial F _c	(semuanya trivial)

2. NORMAL FORM

Tabel	apple_id
Bentuk Normal Form	BCNF
Penjelasan	<p>Candidate Key dan Superkey: id karena $\text{id}^+ = (\text{id}, \text{email}, \text{nama_pengguna}, \text{no_telp_recovery}) = \text{apple_id}$</p> <p>Alasan mengapa tabel ini BCNF adalah id adalah superkey dari tabel apple_id</p>

Tabel	playlist
Bentuk Normal Form	BCNF
Penjelasan	<p>Candidate Key dan Superkey: $\text{playlist_id pengguna_id}$ karena $(\text{playlist_id pengguna_id})^+ = (\text{playlist_id}, \text{pengguna_id}, \text{nama}) = \text{playlist}$</p>

	Alasan mengapa tabel ini BCNF adalah playlist_id pengguna_id adalah superkey dari tabel playlist
--	---

Tabel	isi_playlist
Bentuk Normal Form	BCNF
Penjelasan	Alasan mengapa tabel ini BCNF adalah semua functional dependencynya trivial

Tabel	lagu_produk_komersial
Bentuk Normal Form	BCNF
Penjelasan	Alasan mengapa tabel ini BCNF adalah semua functional dependencynya trivial

Tabel	produk_komersial
Bentuk Normal Form	BCNF
Penjelasan	Candidate Key dan Superkey: id karena $id^+ = (id, judul, artis_id, tipe, genre, tanggal_rilis) = produk_komersial$ Alasan mengapa tabel ini BCNF adalah id adalah superkey dari tabel produk_komersial

Tabel	subscription
Bentuk Normal Form	BCNF
Penjelasan	Candidate Key dan Superkey: subscription_id pengguna_id karena $(subscription_id \text{ pengguna}_id)^+ = (subscription_id, pengguna_id, subscription_plan_jenis, tanggal_subscribe, tanggal_expired, status) = subscription$ Alasan mengapa tabel ini BCNF adalah subscription_id pengguna_id adalah superkey dari tabel subscription

Tabel	lirik
Bentuk Normal Form	BCNF
Penjelasan	Candidate Key dan Superkey: line_lagu_id karena $(line_lagu_id)^+ = (line, lagu_id, text, writer_id) = lirik$ Alasan mengapa tabel ini BCNF adalah line_lagu_id adalah superkey dari tabel lirik

Tabel	kualitas_audio_lagu
Bentuk Normal Form	BCNF
Penjelasan	Alasan mengapa tabel ini BCNF adalah semua functional dependencynya trivial

Tabel	lagu
Bentuk Normal Form	BCNF
Penjelasan	Candidate Key dan Superkey: id karena $id^+ = (id, judul, durasi, tanggal_rilis, artis_id, label_id) = lagu$ Alasan mengapa tabel ini BCNF adalah id adalah superkey dari tabel lagu

Tabel	subscription_plan
Bentuk Normal Form	BCNF
Penjelasan	Candidate Key dan Superkey: jenis karena $jenis^+ = (jenis, harga_per_bulan) = subscription_plan$ Alasan mengapa tabel ini BCNF adalah jenis adalah superkey dari tabel subscription_plan

Tabel	video_musik
-------	-------------

Bentuk Normal Form	BCNF
Penjelasan	<p>Candidate Key dan Superkey: id karena $\text{id}^+ = (\text{id}, \text{judul}, \text{artis_id}, \text{lagu_id}, \text{label_id}, \text{durasi}, \text{tanggal_rilis}) = \text{video_musik}$</p> <p>Alasan mengapa tabel ini BCNF adalah id adalah superkey dari tabel video_musik</p>

Tabel	label
Bentuk Normal Form	BCNF
Penjelasan	<p>Candidate Key dan Superkey: id karena $\text{id}^+ = (\text{id}, \text{nama}, \text{tahun_berdiri}, \text{asal_negara}) = \text{label}$</p> <p>Alasan mengapa tabel ini BCNF adalah id adalah superkey dari tabel label</p>

Tabel	video_extra
Bentuk Normal Form	BCNF
Penjelasan	<p>Candidate Key dan Superkey: id karena $\text{id}^+ = (\text{id}, \text{judul}, \text{artis_id}, \text{label_id}, \text{durasi}, \text{tanggal_rilis}) = \text{video_extra}$</p> <p>Alasan mengapa tabel ini BCNF adalah id adalah superkey dari tabel video_extra</p>

Tabel	host_video_extra
Bentuk Normal Form	BCNF
Penjelasan	Alasan mengapa tabel ini BCNF adalah semua functional dependencinya trivial

3. IMPLEMENTASI INTEGRITY CONSTRAINTS

a. Type Constraints

i. Batasan Nilai Tipe Tahun Berdiri

Diperlukan suatu tipe data `TAHUN_BERDIRI` yang *legal values*-nya adalah bilangan bulat positif. Untuk keperluan implementasi, berhubung dalam MariaDB tidak memungkinkan membuat *user-defined data type* maka penggambaran tipe data tersebut akan diimplementasikan melalui *constraint* `tahun_berdiri_check` yang membuat atribut `tahun_berdiri` pada tabel `label` memiliki domain berupa konsep tipe data `TAHUN_BERDIRI` tersebut.

```
// Notasi kuliah
TYPE TAHUN_BERDIRI POSSREP (INTEGER)
    CONSTRAINT TAHUN_BERDIRI_CHECK (TAHUN_BERDIRI) > 0;

// Implementasi
ALTER TABLE label
ADD CONSTRAINT tahun_berdiri_check CHECK (tahun_berdiri > 0);
```

ii. Batasan Jenis Kualitas Audio

Diperlukan suatu tipe data `KUALITAS_AUDIO` yang membatasi kemungkinan jenis Kualitas Audio yang dapat dimiliki oleh suatu lagu. Terdapat 2 jenis subscription plan yang legal yaitu “Hi-Res Lossless” dan “Dolby Atmos”. Untuk keperluan implementasi, berhubung dalam MariaDB tidak memungkinkan membuat *user-defined data type* maka penggambaran tipe data tersebut akan diimplementasikan melalui *enum* yang akan membatas jenis data yang dapat dimasukkan ke dalam suatu atribut.

```
// Notasi kuliah
TYPE KUALITAS_AUDIO POSSREP (VARCHAR)
    CONSTRAINT KUALITAS_AUDIO IN ('Hi-Res Lossless', 'Dolby
Atmos');

// Implementasi
CREATE TABLE IF NOT EXISTS kualitas_audio_lagu (
    lagu_id INT,
    /* Bagian implementasi */

    kualitas_audio ENUM ("Hi-Res Lossless", "Dolby Atmos"),
    PRIMARY KEY (lagu_id, kualitas_audio),
    FOREIGN KEY (lagu_id) REFERENCES lagu (id)
    ON UPDATE CASCADE
```

```
        ON DELETE RESTRICT  
);
```

b. *Attribute Constraints*

i. **Nilai Legal Untuk Atribut Tabel Apple ID**

Berikut merupakan pendefinisian domain dari masing-masing atribut id, email, password, nama_pengguna, dan no_telepon_recovery, serta implementasinya.

- ID menyatakan INT, NOT NULL, dan AUTO_INCREMENT
- VARCHAR_NOTNULL menyatakan VARCHAR(255) NOT NULL

```
// Notasi kuliah  
VAR apple_id BASE RELATION (  
    id ID,  
    email VARCHAR_NOTNULL,  
    password VARCHAR_NOTNULL,  
    nama_pengguna VARCHAR_NOTNULL,  
    no_telepon_recovery VARCHAR_NOTNULL  
);  
  
// Implementasi (telah diterapkan)  
CREATE TABLE IF NOT EXISTS apple_id (  
    id INT AUTO_INCREMENT,  
    email VARCHAR(255) NOT NULL,  
    password VARCHAR(255) NOT NULL,  
    nama_pengguna VARCHAR(255) NOT NULL,  
    no_telepon_recovery VARCHAR(255) NOT NULL,  
  
    PRIMARY KEY (id)  
);
```

ii. **Nilai Legal Untuk Atribut Nama Tabel Label**

Berikut merupakan pendefinisian domain dari masing-masing atribut id, nama, tahun_berdiri, dan asal_negara.

- ID menyatakan INT, NOT NULL, dan AUTO_INCREMENT
- INT_NOTNULL menyatakan INT NOT NULL
- VARCHAR_NOTNULL menyatakan VARCHAR(255) NOT NULL

```

// Notasi kuliah
VAR label BASE RELATION (
    id ID,
    nama VARCHAR_NOTNULL,
    tahun_berdiri INT_NOTNULL,
    asal_negara VARCHAR_NOTNULL
);

// Implementasi (telah diterapkan)
CREATE TABLE IF NOT EXISTS label (
    id INT AUTO_INCREMENT,
    nama VARCHAR(255) NOT NULL,
    tahun_berdiri INT NOT NULL,
    asal_negara VARCHAR(255) NOT NULL,

    PRIMARY KEY (id)
);

```

c. Relation Constraints

i. Validasi Pembuatan Tanggal Expired

Subscription plan yang dimiliki Apple Music akan berlaku selama satu bulan. Definisi dari satu bulan adalah tanggal yang sama di bulan selanjutnya dengan ketentuan bahwa apabila jumlah hari di bulan selanjutnya lebih sedikit dari bulan sekarang dan tanggal terkait tidak ada di bulan selanjutnya, maka akan diambil tanggal terbesar di bulan selanjutnya. Untuk mencegah adanya pemasukan-pemasukan data yang tidak bertanggung jawab dan cenderung merugikan perusahaan, maka dari itu akan dibuat sebuah trigger ketika data akan di-insert ke tabel *subscription*. Trigger akan melakukan cek apakah data yang akan dimasukkan ke basis data valid atau tidak. Jika sesuai, maka *insert* akan berhasil, jika tidak maka akan *rollback*.

```

// Notasi kuliah
// Tidak boleh ada atribut tanggal_expired yang tidak sesuai
// Constraint yang telah dijelaskan

CONSTRAINT VALIDASI_TANGGAL_EXPIRED
IS_EMPTY (subscription WHERE tanggal_expired <>
DATE_ADD(tanggal_subscribe, INTERVAL 1 MONTH))

CREATE TRIGGER validate_tanggal_expired_before_insert

```

```

BEFORE INSERT OF subscription
REFERENCING NEW ROW AS nrow
    IF nrow.tanggal_expired <> DATE_ADD(nrow.tanggal_subscribe,
INTERVAL 1 MONTH)
        THEN begin
            ROLLBACK TRANSACTION;
        END;

// Implementasi di sql
DROP TRIGGER IF EXISTS validate_tanggal_expired_before_insert;

DELIMITER //

CREATE TRIGGER validate_tanggal_expired_before_insert
BEFORE INSERT ON subscription
FOR EACH ROW
BEGIN
    DECLARE next_month_date DATE;
    SET next_month_date = DATE_ADD(NEW.tanggal_subscribe,
INTERVAL 1 MONTH);

    IF NEW.tanggal_expired != next_month_date THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
'tanggal_expired must be exactly one month from
tanggal_subscribe';
    END IF;
END //

DELIMITER ;

```

ii. Informasi yang lengkap

Untuk memastikan pengalaman terbaik bagi pengguna, diperlukan suatu mekanisme agar semua informasi yang ditampilkan ke pengguna lengkap. Lengkap disini berarti bahwa apabila pengguna ingin melihat informasi mengenai suatu konten, maka semua hal dan informasi umum yang terkait dengan konten tersebut haruslah tersedia. Hal ini dapat dicapai dengan menambahkan *not null* constraint di setiap atribut dari tabel-tabel sebuah konten.

```

// Notasi Kuliah, tidak boleh ada atribut yang null dari konten
// - konten di apple music

CONSTRAINT LAGU
IS_EMPTY(lagu WHERE id IS NULL OR artis_id IS NULL OR label_id IS
NULL OR judul IS NULL OR durasi IS NULL OR tanggal_rilis IS NULL)

CONSTRAINT VIDEO_EXTRA

```

```

IS_EMPTY(video_extra WHERE id IS NULL OR artis_id IS NULL OR
label_id IS NULL OR judul IS NULL OR durasi IS NULL OR
tanggal_rilis IS NULL)

CONSTRAINT VIDEO_MUSIK
IS_EMPTY(video_musik WHERE id IS NULL OR artis_id IS NULL OR
label_id IS NULL OR judul IS NULL OR durasi IS NULL OR
tanggal_rilis IS NULL OR lagu_id IS NULL)

/*
Pastikan seluruh entry dari tabel lagu atributnya tidak ada
yang null. Telah diterapkan */

CREATE TABLE IF NOT EXISTS lagu (
    id                  INT AUTO_INCREMENT,
    artis_id            INT NOT NULL,
    label_id            INT NOT NULL,
    judul               VARCHAR(255) NOT NULL,
    durasi              INT NOT NULL,
    tanggal_rilis      DATE NOT NULL,

    PRIMARY KEY (id),

    /* Abaikan bagian dibawah ini, karena tidak terkait dengan
    not null constraint yang sedang dibahas, namun diperlukan untuk
    syntax sql */
    FOREIGN KEY (artis_id) REFERENCES apple_id (id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    FOREIGN KEY (label_id) REFERENCES label (id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

/*
Pastikan seluruh entry dari tabel video_extra atributnya tidak
ada yang null. Telah diterapkan */

CREATE TABLE IF NOT EXISTS video_extra (
    id                  INT AUTO_INCREMENT,
    artis_id            INT NOT NULL,
    label_id            INT NOT NULL,
    judul               VARCHAR(255) NOT NULL,
    durasi              INT NOT NULL,
    tanggal_rilis      DATE NOT NULL,

    /* Abaikan bagian dibawah ini, karena tidak terkait dengan
    not null constraint yang sedang dibahas, namun diperlukan untuk
    syntax sql */

    PRIMARY KEY (id),
    FOREIGN KEY (artis_id) REFERENCES apple_id (id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    FOREIGN KEY (label_id) REFERENCES label (id)

```

```

        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

/* Pastikan seluruh entry dari tabel video_musik atributnya tidak
ada yang null. Telah diterapkan */

CREATE TABLE IF NOT EXISTS video_musik (
    id                  INT AUTO_INCREMENT,
    artis_id            INT NOT NULL,
    lagu_id              INT NOT NULL,
    label_id             INT NOT NULL,
    judul                VARCHAR(255) NOT NULL,
    durasi               INT NOT NULL,
    tanggal_rilis        DATE NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (artis_id) REFERENCES apple_id (id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    FOREIGN KEY (lagu_id) REFERENCES lagu (id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    FOREIGN KEY (label_id) REFERENCES label (id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

```

d. Database Constraints

i. Artis dari lagu dalam Produk Komersial

Pada spesifikasi, disebutkan bahwa artis dari lagu-lagu yang ada di dalam suatu produk komersial haruslah sama dengan artis yang ada pada produk komersial terkait. Hal ini dapat dicapai dengan membuat *trigger* ketika akan ada suatu lagu yang dimasukkan ke dalam tabel lagu_produk_komersial, maupun ketika akan ada suatu perubahan yang dilakukan pada tabel terkait. Perubahan data yang diizinkan hanya apabila artis yang akan dimasukkan sama dengan artis pada produk komersial terkait.

```
// Notasi Kuliah, tidak boleh ada lagu di dalam produk komersial
```

```

// yang pembuatnya bukan artis di produk komersial

CONSTRAINT LAGU_PRODUK_KOMERSIAL
IS_EMPTY (
    (lagu_produk_komersial as lpk join lagu as l on l.id =
lpk.lagu_id join produk_komersial as p on p.id =
lpk.produk_komersial_id)
    WHERE lagu.artis_id <> produk_komersial.artis_id
)

// Implementasi trigger dengan notasi perkuliahan
CREATE TRIGGER check_artis_id
BEFORE INSERT OF lagu_produk_komersial
REFERENCING NEW ROW AS nrow
IF EXISTS (
    SELECT 1
    FROM produk_komersial
    WHERE id = nrow.produk_komersial_id
    AND artis_id <> (
        SELECT artis_id
        FROM lagu
        WHERE id = nrow.lagu_id
    )
) THEN begin
    ROLLBACK TRANSACTION;
END;

CREATE TRIGGER check_artis_id_update
BEFORE UPDATE OF lagu_produk_komersial
REFERENCING NEW ROW AS nrow
IF EXISTS (
    SELECT 1
    FROM produk_komersial
    WHERE id = nrow.produk_komersial_id
    AND artis_id <> (
        SELECT artis_id
        FROM lagu
        WHERE id = nrow.lagu_id
    )
) THEN begin
    ROLLBACK TRANSACTION;
END;

// Implementasi trigger sebelum insert di sql
DROP TRIGGER IF EXISTS check_artis_id;

DELIMITER //

CREATE TRIGGER check_artis_id
BEFORE INSERT ON lagu_produk_komersial
FOR EACH ROW
BEGIN
    DECLARE v_artis_id_produk_komersial INT;
    DECLARE v_artis_id_lagu INT;

```

```

SELECT artis_id INTO v_artis_id_produk_komersial
FROM produk_komersial
WHERE id = NEW.produk_komersial_id;

SELECT artis_id INTO v_artis_id_lagu
FROM lagu
WHERE id = NEW.lagu_id;

IF v_artis_id_produk_komersial != v_artis_id_lagu THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'The artist of the product must match
the artist of the song.';
END IF;
END //

DELIMITER ;

/* Implementasi trigger sebelum update */
DROP TRIGGER IF EXISTS check_artis_id_before_update;

DELIMITER //

CREATE TRIGGER check_artis_id_before_update
BEFORE UPDATE ON lagu_produk_komersial
FOR EACH ROW
BEGIN
    DECLARE v_artis_id_produk_komersial INT;
    DECLARE v_artis_id_lagu INT;

    SELECT artis_id INTO v_artis_id_produk_komersial
    FROM produk_komersial
    WHERE id = NEW.produk_komersial_id;

    SELECT artis_id INTO v_artis_id_lagu
    FROM lagu
    WHERE id = NEW.lagu_id;

    -- Check if artis_id of the product matches artis_id of the
    -- song
    IF v_artis_id_produk_komersial != v_artis_id_lagu THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The artist of the product must match
the artist of the song.';
    END IF;
END //

DELIMITER ;

```

ii. Tanggal Rilis Video Musik

Pada spesifikasi, disebutkan bahwa suatu video musik harus terikat dengan satu lagu. Pada dunia nyata, video musik pastilah dirilis setelah sebuah lagu dirilis, atau bersamaan dengan lagu tersebut dirilis. Menjadi tidak masuk akal apabila kemudian ada suatu video musik yang mendahului rilis dari lagunya. Maka dari itu, basis data perlu disesuaikan untuk memastikan ketika ada data video musik yang ingin diasosiasikan pada sebuah lagu, *constraint* terkait dipenuhi. Hal ini dapat diimplementasikan dengan *trigger* yang akan menghandle *event insert* pada tabel `video_musik`. Apabila data yang ingin diinput valid (tanggalnya setelah atau sama dengan tanggal dari lagu terkait) maka operasi jadi dilakukan. Apabila tidak, maka akan di *rollback*.

```
// Notasi Kuliah, tidak boleh ada video musik yang rilis sebelum
// lagu yang terkait dengannya

CONSTRAINT VID_MUSIC_LAGU
IS_EMPTY (
    (video_musik JOIN lagu on video_musik.lagu_id = lagu.id)
WHERE video_musik.tanggal_rilis < lagu.tanggal_rilis
)

CREATE TRIGGER check_video_release_date
BEFORE INSERT OF video_musik
REFERENCING NEW ROW AS nrow
    IF EXISTS (
        SELECT 1
        FROM lagu
        WHERE id = nrow.lagu_id
        AND nrow.tanggal_rilis < tanggal_liris)
    ) THEN begin
        ROLLBACK TRANSACTION;
    END;

// Implementasi DI SQL
DROP TRIGGER IF EXISTS check_video_release_date;

DELIMITER //

CREATE TRIGGER check_video_release_date
BEFORE INSERT ON video_musik
FOR EACH ROW
BEGIN
    DECLARE lagu_release_date DATE;

    SELECT tanggal_rilis INTO lagu_release_date
    FROM lagu
    WHERE id = NEW.lagu_id;
```

```

-- Check if the video release date is before the lagu release
date
IF NEW.tanggal_rilis < lagu_release_date THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Video release
date must be at least the same or after the release date of the
corresponding lagu';
END IF;
END // 

DELIMITER ;

```

e. *Transition Constraints*

i. Perubahan Status Langganan Menjadi "Inaktif"

Pada tabel ***subscription***, atribut ***status*** menunjukkan status langganan pengguna aktif atau tidak aktif. Constraint ini memastikan perubahan status langganan ***subscription*** dari “aktif” menjadi “inaktif” jika tanggal saat ini melebihi ***tanggal_expired*** yang dimasukkan. Strategi yang digunakan adalah dengan trigger yang memperbarui status langganan menjadi “inaktif” jika ***tanggal_expired*** lebih dari tanggal saat ini.

```

// Notasi Kuliah
// NEW_SUBSCRIPTION menandakan hasil perubahan data
CONSTRAINT TRC_ValidStatus IS_EMPTY
( ( NEW_SUBSCRIPTION (subscription_id, pengguna_id,
tanggal_expired, status) RENAME tanggal_expired AS
new_tanggal_expired, status AS new_status)
    JOIN SUBSCRIPTION (subscription_id, pengguna_id,
tanggal_expired, status)
    WHERE new_status = 'aktif' AND new_tanggal_expired <
CURRENT_DATE() );

// Implementasi SQL

DROP TRIGGER IF EXISTS update_subscription_status;

DELIMITER $$

CREATE TRIGGER update_subscription_status
BEFORE UPDATE ON subscription
FOR EACH ROW
BEGIN
    IF NEW.tanggal_expired < CURRENT_DATE() THEN
        SET NEW.status = 'inaktif';
    END IF;
END $$

DELIMITER ;

```

ii. Perubahan jenis subscription plan

Pada tabel `subscription`, atribut `subscription_plan_jenis` menunjukkan jenis paket langganan yang dimiliki oleh pengguna saat ini. Maka dari itu, apabila ingin diadakan perubahan di jenis subscription plan yang ada, transition yang valid adalah apabila masih diantara 3 jenis Subscription valid yaitu “Pelajar”, “Keluarga” dan “Perorangan”. Constraint ini memastikan perubahan jenis langganan tetap berada di nilai-nilai yang valid. Strategi yang digunakan adalah dengan membuat sebuah enum values dari subscription plan sehingga perubahan pastilah valid.

```
// Notasi Kuliah
CONSTRAINT TRC_VALIDATE_SUBSCRIPTION IS_EMPTY
( ( NEW_SUBSCRIPTION (subscription_id, pengguna_id,
tanggal_expired, status) RENAME tanggal_expired AS
new_tanggal_expired, status AS new_status)
JOIN SUBSCRIPTION (subscription_id, pengguna_id,
tanggal_expired, status)
WHERE new_status = 'aktif' AND new_tanggal_expired <
CURRENT_DATE() ) ;

// Implementasi SQL

CREATE TABLE IF NOT EXISTS subscription (
subscription_id INT,
pengguna_id INT,
subscription_plan_jenis ENUM ("Pelajar", "Perorangan",
"Keluarga"),
tanggal_subscribe DATE NOT NULL,
tanggal_expired DATE NOT NULL,
status ENUM ("aktif", "inaktif"),

PRIMARY KEY (subscription_id, pengguna_id),
FOREIGN KEY (pengguna_id) REFERENCES apple_id (id)
ON UPDATE CASCADE
ON DELETE RESTRICT,
FOREIGN KEY (subscription_plan_jenis) REFERENCES
subscription_plan (jenis)
ON UPDATE CASCADE
ON DELETE RESTRICT
);
```

f. Referential Actions

i. Validasi Pengguna pada Tabel Subscription

Setiap `subscription` harus dihubungkan dengan `pengguna_id` yang valid dari `apple_id` yang akan memastikan keakuratan relasi antara

tabel melalui penggunaan *foreign key*. Constraint ini untuk mencegah adanya “data yatim”, jika

```
CREATE TABLE IF NOT EXISTS `subscription` (
    /* Abaikan sebagian statements berikut */

    `subscription_id` int(11) NOT NULL,
    `pengguna_id` int(11) NOT NULL,
    `subscription_plan_jenis` enum('Pelajar','Perorangan','Keluarga') DEFAULT NULL,
    `tanggal_subscribe` date NOT NULL,
    `tanggal_expired` date NOT NULL,
    `status` enum('aktif','inaktif') DEFAULT NULL,
    PRIMARY KEY (`subscription_id`,`pengguna_id`),
    KEY `pengguna_id` (`pengguna_id`),
    KEY `subscription_plan_jenis` (`subscription_plan_jenis`),

    /* Bagian bawah merupakan referential actions */

    CONSTRAINT `subscription_ibfk_1` FOREIGN KEY (`pengguna_id`)
        REFERENCES `apple_id` (`id`) ON UPDATE CASCADE,
    CONSTRAINT `subscription_ibfk_2` FOREIGN KEY
        (`subscription_plan_jenis`) REFERENCES `subscription_plan`(`jenis`)
        ON UPDATE CASCADE
);
```

ii. Validasi Artis pada Tabel Lagu

Setiap lagu harus memiliki ***artis_id*** yang valid, merujuk pada entri yang ada di tabel ***apple_id***. Dilakukan validasi untuk memastikan bahwa setiap lagu memiliki artis yang valid dan menghindari penghapusan artis yang masih memiliki lagu terkait.

```
CREATE TABLE IF NOT EXISTS `lagu` (
    /* Abaikan sebagian statements berikut */

    `id` int(11) NOT NULL AUTO_INCREMENT,
    `artis_id` int(11) NOT NULL,
    `label_id` int(11) NOT NULL,
    `judul` varchar(255) NOT NULL,
    `durasi` int(11) NOT NULL,
    `tanggal_rilis` date NOT NULL,
    PRIMARY KEY (`id`),
    KEY `artis_id` (`artis_id`),
    KEY `label_id` (`label_id`),

    /* Bagian bawah merupakan referential actions */

    CONSTRAINT `lagu_ibfk_1` FOREIGN KEY (`artis_id`) REFERENCES
        `apple_id` (`id`) ON UPDATE CASCADE,
```

```

CONSTRAINT `lagu_ibfk_2` FOREIGN KEY (`label_id`) REFERENCES
`label` (`id`) ON UPDATE CASCADE
);

```

4. FITUR TAMBAHAN

a. Validasi Pengguna Apple Music

Tujuan	Melakukan validasi atribut artis setiap kali melakukan <i>insertion</i> pada tabel lagu, video musik, dan video ekstra sehingga <i>value</i> artis pada <i>tuple</i> data tambahan dipastikan merupakan Apple ID yang memiliki subscription Apple Music aktif. (atribut hanya dicek saat <i>insertion</i> saja).
Query Testing	<pre> /* Inisialisasi session variable `new_max_lagu_id` yang merupakan nilai `id` dari `lagu` yang akan di-insert. `id` tersebut berperan penting dalam penghapusan penambahan record sebelum trigger didefinisikan */ SET @new_max_lagu_id = (SELECT MAX(id) + 1 FROM lagu); /* Insert sebuah `lagu` dengan `artis_id` yang tidak pernah berlangganan atau status berlangganannya tidak aktif */ INSERT INTO lagu (id, artis_id, label_id, judul, durasi, tanggal_rilis) (SELECT @new_max_lagu_id, aid.min_invalid_artis_id, lb.min_id, 'Dummy title', 120, '2024-01-01' FROM (</pre>

```

SELECT
    MIN(aid.id) AS
    min_invalid_artis_id
FROM (
    (
        SELECT
            id
        FROM
            apple_id
    )
EXCEPT
(
    SELECT DISTINCT
        penguna_id AS
        id
    FROM
        subscription
    WHERE
        status = 'aktif'
)
) AS aid
) AS aid,
(
    SELECT
        MIN(id) AS min_id
    FROM
        label
    ) AS lb
);

/* Retrieve `lagu` yang dicoba untuk di-insert */

SELECT
    *
FROM
    lagu
WHERE
    id = @new_max_lagu_id;

/* Hapus `lagu` yang dicoba untuk di-insert tersebut
in-case peng-insert-an berhasil */

DELETE
FROM
    lagu
WHERE
    id = @new_max_lagu_id;

/* Retrieve lagu yang `id`-nya maksimum.
Ekspektasinya adalah data lama */

SELECT
    *
FROM
    lagu
ORDER BY

```

	<pre> id DESC LIMIT 1; </pre>
Query Manipulasi	<pre> /* Buat function `is_apple_id_active_subscribing` untuk menentukan apakah benar Apple ID `id` berstatus aktif berlangganan */ DROP FUNCTION IF EXISTS is_apple_id_active_subscribing; DELIMITER \$\$ CREATE FUNCTION is_apple_id_active_subscribing (id INT) RETURNS BOOLEAN BEGIN DECLARE is_subscribe BOOLEAN; SELECT EXISTS (SELECT pengguna_id FROM subscription WHERE pengguna_id = id AND status = 'aktif') INTO is_subscribe; RETURN is_subscribe; END \$\$ DELIMITER ; /* Buat trigger untuk masing-masing tabel lagu, video musik, dan video extra sesuai yang tertulis pada tujuan */ DROP TRIGGER IF EXISTS before_insert_lagu; DELIMITER \$\$ CREATE TRIGGER before_insert_lagu BEFORE INSERT ON lagu FOR EACH ROW BEGIN IF NOT (is_apple_id_active_subscribing(NEW.artis_id)) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Artis ID tidak valid karena belum pernah berlangganan atau status berlangganannya tidak aktif'; END IF; END \$\$ </pre>

DELIMITER ;

```

MariaDB [tubes_lf2240_apple_music]> /* Retrieve 'lagu' yang dicoba untuk di-insert */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> SELECT
-- *
-- FROM
-- lagu
-- WHERE
-- > id = @new_max_lagu_id;
+-----+
| id | artis_id | tabel_id | judul | durasi | tanggal_rilis |
+-----+
| 151 |      51 |        1 | Dummy title |   120 | 2024-01-01 |
+-----+
1 row in set (0,000 sec)

MariaDB [tubes_lf2240_apple_music]> /* Hapus 'lagu' yang dicoba untuk di-insert tersebut in-case peng-insert-an berhasil */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DELETE
-- FROM
-- lagu
-- WHERE
-- > id = @new_max_lagu_id;
Query OK, 1 row affected (0,014 sec)

MariaDB [tubes_lf2240_apple_music]> /* Retrieve lagu yang 'id'-nya maksimum. Ekspektasinya adalah data lama */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> SELECT
-- *
-- FROM
-- lagu
-- ORDER BY
-- > id DESC
-- LIMIT 1;
+-----+
| id | artis_id | tabel_id | judul | durasi | tanggal_rilis |
+-----+
| 150 |      11 |       83 | This Land Is Your Land |    278 | 2020-03-16 |
+-----+
-- *
-- FROM
-- lagu
-- WHERE
-- > id = @new_max_lagu_id;
+-----+
| id | artis_id | tabel_id | judul | durasi | tanggal_rilis |
+-----+
| 151 |      51 |        1 | Dummy title |   120 | 2024-01-01 |
+-----+
1 row in set (0,000 sec)

MariaDB [tubes_lf2240_apple_music]> /* Hapus 'lagu' yang dicoba untuk di-insert tersebut in-case peng-insert-an berhasil */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DELETE
-- FROM
-- lagu
-- WHERE
-- > id = @new_max_lagu_id;
Query OK, 1 row affected (0,014 sec)

MariaDB [tubes_lf2240_apple_music]> /* Retrieve lagu yang 'id'-nya maksimum. Ekspektasinya adalah data lama */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> SELECT
-- *
-- FROM
-- lagu
-- ORDER BY
-- > id DESC
-- LIMIT 1;
+-----+
| id | artis_id | tabel_id | judul | durasi | tanggal_rilis |
+-----+
| 150 |      11 |       83 | This Land Is Your Land |    278 | 2020-03-16 |
+-----+
1 row in set (0,000 sec)

MariaDB [tubes_lf2240_apple_music]> ■

```

Tangkapan Layar Proses Manipulasi

```

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /* Buat function 'is_apple_id_active_subscribing' untuk menentukan apakah benar Apple ID 'id' berstatus aktif berlangganan */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DROP FUNCTION IF EXISTS is_apple_id_active_subscribing;
Query OK, 0 rows affected, 1 warning (0,068 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DELIMITER $$
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> CREATE FUNCTION is_apple_id_active_subscribing (
-- > id INT
-- > )
-- > RETURNS BOOLEAN
-- > BEGIN
-- > DECLARE ls_subscribe BOOLEAN;
-- >
-- > SELECT
-- > EXISTS (
-- > SELECT
-- > pengguna_id
-- > FROM
-- > subscription
-- > WHERE
-- > pengguna_id = id
-- > AND status = 'aktif'
-- > ) INTO ls_subscribe;
-- >
-- > RETURN ls_subscribe;
-- > END $$
Query OK, 0 rows affected (0,406 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DELIMITER ;
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /* Buat trigger untuk masing-masing tabel lagu, video musik, dan video extra sesuai yang tertulis pada tujuan */
MariaDB [tubes_lf2240_apple_music]> 

```

```
--> SELECT
--> pengguna_id
--> FROM
--> subscription
--> WHERE
--> pengguna_id = id
--> AND status = 'aktif'
--> ) INTO is_subscribe;
-->
--> RETURN is_subscribe;
--> END $$

Query OK, 0 rows affected (0,406 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELIMITER ;
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Buat trigger untuk masing-masing tabel lagu, video musik, dan video extra sesuai yang tertulis pada tujuan */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> MariaDB [tubes_if2240_apple_music]> DROP TRIGGER IF EXISTS before_insert_lagu;
Query OK, 0 rows affected, 1 warning (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELIMITER $$

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> CREATE TRIGGER before_insert_lagu
--> BEFORE INSERT ON lagu
--> FOR EACH ROW
--> BEGIN
--> IF NOT (is_apple_id_active_subscribing(NEW.artis_id)) THEN
--> SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Artis ID tidak valid karena belum pernah berlangganan atau status berlanggannya tidak aktif';
--> END IF;
--> END $$

Query OK, 0 rows affected (0,165 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELIMITER ;
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> █
```

Tangkapan Layar Setelah Manipulasi (Ekspektasi insertion ditolak)

```
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Inisialisasi session variable `new_max_lagu_id` yang merupakan nilai 'id' dari 'lagu' yang akan di-insert. 'id' tersebut berperan penting dalam penghapusan penambahan record sebelum trigger didefinisikan */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> SET @new_max_lagu_id = (
->   SELECT
->     MAX(id) + 1
->   FROM
->     lagu
-> );
Query OK, 0 rows affected (0,001 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Insert sebuah 'lagu' dengan 'artis_id' yang tidak pernah berlangganan atau status berlangganannya tidak aktif */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> INSERT INTO lagu (
->   id,
->   artis_id,
->   label_id,
->   judul,
->   durasi,
->   tanggal_rilis
-> ) (
->   SELECT
->     @new_max_lagu_id,
->     aid.min_invalid_artis_id,
->     lb.min_id,
->     'Dummy title',
->     120,
->     '2024-01-01'
->   FROM
->   (
->     SELECT
->       MIN(aid.id) AS min_invalid_artis_id
->     FROM (
->       (
->         SELECT
->           aid,
->           aid.min_invalid_artis_id,
->           lb.min_id,
->           'Dummy title',
->           120,
->           '2024-01-01'
->         FROM
->         (
->           SELECT
->             MIN(aid.id) AS min_invalid_artis_id
->           FROM (
->             (
->               SELECT
->                 aid,
->                 aid.id
->               FROM
->                 apple_id
->             )
->             EXCEPT
->             (
->               SELECT DISTINCT
->                 pengguna_id AS id
->               FROM
->                 subscription
->               WHERE
->                 status = 'aktif'
->             )
->             ) AS aid
->             ) AS aid,
->             (
->               SELECT
->                 MIN(id) AS min_id
->               FROM
->                 label
->                 AS lb
->               )
->             )
->           )
->         )
->       )
->     )
->   )
-> );
Query OK, 1 row affected (0,001 sec)

MariaDB [tubes_if2240_apple_music]> SELECT * FROM lagu
+----+-----+-----+-----+-----+-----+
| id | artis_id | label_id | judul | durasi | tanggal_rilis |
+----+-----+-----+-----+-----+-----+
| 121 |          |          |        |      120 | 2024-01-01    |
+----+-----+-----+-----+-----+-----+
1 row in set (0,001 sec)
```

```

MariaDB [tubes_lf2240_apple_music]> /* Retrieve 'lagu' yang dicoba untuk di-insert */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> SELECT
-> *
-> FROM
-> lagu
-> WHERE
-> id = @new_max_lagu_id;
Empty set (0,000 sec)

MariaDB [tubes_lf2240_apple_music]> /* Hapus 'lagu' yang dicoba untuk di-insert tersebut in-case peng-insert-an berhasil */
MariaDB [tubes_lf2240_apple_music]> DELETE
-> FROM
-> lagu
-> WHERE
-> id = @new_max_lagu_id;
Query OK, 0 rows affected (0,000 sec)

MariaDB [tubes_lf2240_apple_music]> /* Retrieve lagu yang 'id'-nya maksimum. Ekspektasinya adalah data lama */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> SELECT
-> *
-> FROM
-> lagu
-> ORDER BY
-> id DESC
-> LIMIT 1;
+-----+-----+-----+-----+
| id | artis_id | label_id | judul           | durasi | tanggal_rilis |
+-----+-----+-----+-----+
| 150 |        11 |       83 | This Land Is Your Land |    278 | 2020-03-16   |
+-----+-----+-----+-----+
1 row in set (0,000 sec)

```

b. *Multiple Songwriter*

(Tabel bisa disesuaikan, namun minimal ada ini). Semua persoalan di fitur tambahan **wajib diimplementasikan** pada **basis data**.

Tujuan	Membuat sebuah <i>view</i> yang berisikan identitas lagu dan liriknya secara keseluruhan (cukup pisahkan dengan spasi saja), diikuti dengan atribut <i>writers</i> yang berisikan seluruh <i>writer</i> (unik) dari lagu tersebut yang dipisahkan oleh tanda koma.
Query Testing	<pre> SHOW FULL TABLES LIKE 'multiple_songwriter'; DESC multiple_songwriter; SELECT * FROM multiple_songwriter\G </pre>
Query Manipulasi	<pre> /* Berhubung pada instance database saat ini setiap lagu yang memiliki lirik, seluruh liriknya hanya dibuat oleh seorang writer, maka akan ditambahkan beberapa dummy data untuk dimasukkan sebagai pembanding */ INSERT INTO lirik VALUES (11, 20, 1, 'Ini lirik lagu tambahan dari writer yang berbeda'); INSERT INTO lirik VALUES (12, 20, 2, 'Ini lirik lagu tambahan dari writer yang berbeda lagi'); </pre>

```

DROP VIEW IF EXISTS multiple_songwriter;

CREATE VIEW multiple_songwriter AS (
    SELECT
        l.lagu_id,
        GROUP_CONCAT(l.text SEPARATOR ' ') AS
            content,
        GROUP_CONCAT(DISTINCT aid.nama_pengguna
SEPARATOR ', ') AS writers
    FROM
        lirik AS l
    INNER JOIN apple_id AS aid
        ON (l.writer_id = aid.id)
    GROUP BY
        l.lagu_id
);

```

Tangkapan Layar Sebelum Manipulasi

```

MariaDB [tubes_if2240_apple_music]> SHOW FULL TABLES LIKE 'multiple_songwriter';
Empty set (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DESC multiple_songwriter;
ERROR 1146 (42S02): Table 'tubes_if2240_apple_music.multiple_songwriter' doesn't exist
MariaDB [tubes_if2240_apple_music]> SELECT
-- *
-- > FROM
-- > multiple_songwriter\G
ERROR 1146 (42S02): Table 'tubes_if2240_apple_music.multiple_songwriter' doesn't exist
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> █

```

Tangkapan Layar Proses Manipulasi

```

MariaDB [tubes_if2240_apple_music]> /* Berhubung pada instance database saat ini setiap lagu yang memiliki lirik, seluruh liriknya hanya dibuat oleh seorang writer, maka akan ditambahkan beberapa dummy data untuk dimasukkan sebagai pembanding */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> INSERT INTO lirik VALUES (11, 20, 1, 'Ini lirik lagu tambahan dari writer yang berbeda');
Query OK, 1 row affected (0,071 sec)

MariaDB [tubes_if2240_apple_music]> INSERT INTO lirik VALUES (12, 20, 2, 'Ini lirik lagu tambahan dari writer yang berbeda lagi');
Query OK, 1 row affected (0,010 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DROP VIEW IF EXISTS multiple_songwriter;
Query OK, 0 rows affected (0,074 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> CREATE VIEW multiple_songwriter AS (
    > SELECT
    >     l.lagu_id,
    >     GROUP_CONCAT(l.text SEPARATOR ' ') AS content,
    >     GROUP_CONCAT(DISTINCT aid.nama_pengguna SEPARATOR ', ') AS writers
    >     FROM
    >     lirik AS l
    >     INNER JOIN apple_id AS aid
    >     ON (l.writer_id = aid.id)
    >     GROUP BY
    >     l.lagu_id
    > );
Query OK, 0 rows affected (0,084 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]>

```

Tangkapan Layar Setelah Manipulasi

```

MariaDB [tubes_if2240_apple_music]> SHOW FULL TABLES LIKE 'multiple_songwriter';
+-----+-----+
| Tables_in_tubes_if2240_apple_music | Table_type |
+-----+-----+
| multiple_songwriter                | VIEW       |
+-----+-----+
1 row in set (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DESC multiple_songwriter;
+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| lagu_id | int(11) | NO   |   | NULL    |          |
| content | mediumtext | YES  |   | NULL    |          |
| writers | mediumtext | YES  |   | NULL    |          |
+-----+-----+-----+-----+
3 rows in set (0,051 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> SELECT
    > *
    > FROM
    > multiple_songwriter\G
***** 1. row *****
lagu_id: 1
content: Uredo artificioso amittit cur adusesco aperiam aliquia audax comptus. Xiphias exortationem cunctatio comprehendo corrigo vicinus denuncio. St ips sordes uterque valeo crebet peccatus. Aduro vestrum stabilis sortitus vitae porro voluntarius. Magnam atrox absum depulso cuppedia cauda. Adusesco quis causa attollo adfictio abbas tenuis cunque urbanus. Scolius agnoscere ciuis acerbitas thorax matores suppono. Stips velum clementia sufficio tergum cras. Agnosco vorago tui attonbitus aperiam. Callide causus cribro aliquia in beatae succedo cedo.
writers: Haryanto Jarwl
***** 2. row *****
lagu_id: 2
content: Explicabo ratione trans color totam triduana. Totidem benigne sulum stabilis reiciendis. Damnatio territo bardus trado eaque ademptio beneficium. Suppellex degenero tenetur astrum adipisci. Depono compello vester. Nemo verus aer officia modi vereor deputo perferendis umerus acer. Una arca u
    > *
    > FROM
    > multiple_songwriter\G
***** 1. row *****
lagu_id: 1
content: Uredo artificioso amittit cur adusesco aperiam aliquia audax comptus. Xiphias exortationem cunctatio comprehendo corrigo vicinus denuncio. St ips sordes uterque valeo crebet peccatus. Aduro vestrum stabilis sortitus vitae porro voluntarius. Magnam atrox absum depulso cuppedia cauda. Adusesco quis causa attollo adfictio abbas tenuis cunque urbanus. Scolius agnoscere ciuis acerbitas thorax matores suppono. Stips velum clementia sufficio tergum cras. Agnosco vorago tui attonbitus aperiam. Callide causus cribro aliquia in beatae succedo cedo.
writers: Haryanto Jarwl
***** 2. row *****
lagu_id: 2
content: Explicabo ratione trans color totam triduana. Totidem benigne sulum stabilis reiciendis. Damnatio territo bardus trado eaque ademptio beneficium. Suppellex degenero tenetur astrum adipisci. Depono compello vester. Nemo verus aer officia modi vereor deputo perferendis umerus acer. Una arca u
merus coadunato dignissimos tergo surculus. Ter tamistum barba spargo depono tonsor thermae. Ambulo arguo votum asper ocer inflammatio. Utrinque con
or complectus tonsor arx depulso cubo vel non.
writers: Irawan Rathan
***** 3. row *****
lagu_id: 3
content: Utilis odio stultus. Reprehenderit sustineo tolero votum accusantium tego. Consequuntur solvo audax pecus apostolus decipio sufficio. Rerum t
ametsi tollo distinctio aegrus fugit admoneo assumenda. Desparatus suscipit corona subiecto depraeor alt sulum videlicet libero provident. Cavus ambu
lo acceptus. Careo tandem argumentum veritatis subiecto claro asper. Batulus vilicus absens nulla corpus cursus denique tabigo colo alveus. Parlatur cl
audio defetiscor addo vulnifer vesco claudeo sumo cognatus. Animus fugiat paens ab paulatim nesciunt civis validus nulla arbitrio.
writers: Sinaga Dartomo
***** 4. row *****
lagu_id: 4
content: Brevis super patior carcer conventus aveho. Tres curis sint. Aduro depono comptus curiositas videlicet esse. Vespllio tollo curia. Laborum c
opus vento quasi et. Creptio acced aetas qui stultus animadverto. Amoveo repellendus correptus carbo possimus tempus contra compello antiquis conte
cto. Canonicus uredio collum una sit autem. Aranea denuncio despicio tutamen tabernus sui varletas tot aggtero. Ustulo ciminatio sumo angustus accendo i
ure
writers: Ghani Ghani Adriansyah
***** 5. row *****
lagu_id: 5
content: Calco comminor cibus civis adicio. Cauda provident annus pecco amo ventito modi aptus. Crur praesentium canto patria vociferor ademptio moles
tias cur eligendi careo. Quas cubicularis tantillus. Atqui animus appello. Celer solus subvenio. Debilito vociferor uredio casus veniam ademptio non sum
mopere antiquus. Theologus ad aut cubo tumultus impedit curriculum decumbo aequus. Veniam bellum ea alias talis. Tristis beatus sumnopere delinquo.
writers: Kahyang Septi
***** 6. row *****

```

```

content: Agnosco suscipio antea comminor quae de You can paste the image from the clipboard. irbo carbo esse cerno caveo deleo. Sum caecus t
halassinus teres virga eum colo. Trepide sub cet. absco solium cognatus sit totus. Audax absens
absens ulterius appositus curso bonus cubo. Alienus varietas deputo terreo adduco tripudio ulciscor. Corroboro nostrum perforendis textus. Sut crudeli
s ver maxime theca contra comprehendeo. Alveus balbus coaegresco carbo paulatim suasoria molestiae consequuntur.
writers: Drajat Lazuardi *****
***** 17. row *****
lagu_id: 17
content: Defero ultra stillicidium assumenta cupto corporis dolore cibus. Calculus amo coept trucidu solium astrum vtiae vesper demulceo. Decretum dec
erne dolores vir aptus villa brevis non crapula. Cauta bestia censura amito modi bis deputo clarus. Tergiversatio vulariter ante tolero. Cribro suppo
ne ratione nobis. Quod acerbitas sustineo demo tracto collecto. Inventore cum declinus communis veritas. Calculus tantillus voluptatem tempore valens a
rchitecto possimus auditor vulticulos. Ciminatio minima patrocinor supplex tersus hic super subseco alius.
writers: Irawan Rathan *****
***** 18. row *****
lagu_id: 18
content: Pel utique rerum vincio confido sto angulus. Cui ventito addidus occaecati conculco. Consequatur autus absens distinctio desino vox aestivus c
orepitus thymus. Autem volubilis occaecati desigo surgo spero voluntarius volva bardus. Conatus subseco alt ducimus incidunt. Depulso articulus pel
caris. Quam ulterius nam vinum adipisci collecto uberrime aliquia. Stabilis tensor utique credo torqueo Molitia spiculum velociter at summa. Vapulus
admirato totidem tibi benigna utroque acer claustrum tremo. Tandem tenus vos aeternus cilicium.
writers: Syahreza Ivan *****
***** 19. row *****
lagu_id: 19
content: Culterellus congregatio suadeo appello sequi certus. Modi dicta tollo spero tenetur ceno tego umbra. Cribro urbs ultra claustrum valeo. Adhaero
surgo tepidus suppono curro amo coniecto. Animi celo adeo desigo defetiscor amicitia aestas. Accendo concedo depresso vereor stips assumenda sordeo ae
ger. Recusandae vestigium a laborum universe surculus. Pecto bonus venustas delectus sequi desino torqueo. Infit caveo deripio ipsum aurum contigo sor
deo. Tonsor vestigium summisce cubo derideo suffoco cogito surculus.
writers: Vera Yolanda *****
***** 20. row *****
lagu_id: 20
content: Ustilo cubo centum aeternus coma. Pecto denuncio qui voluptate dolore acceptus adulatio. Ini lirik lagu tambahan dari writer yang berbeda Des
iplo capto cunabula alo placeat ater tergeo amplus carlos arcus. Virga curiositas eaque temptatio adflectus demo. Cognomen derelinguo alias timidus s
ollo ex. Necesstitibus tardus pel. Sopor sperno angulus comparo deporto derideo cicut terra. Sodalitas possimus aperiam quod armarium uxori comprehen
do. Territo trucido vociferor anser stipes minima praesentium suscipit canonicus. Provident sto ver desolo fuga aestas termes decens. Ini lirik lagu t
ambahan dari writer yang berbeda lagi
writers: Ajeng Salsabilia, Haryanto Jarwi, Irawan Rathan
20 rows in set (0,002 sec)
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> ■

```

c. Statistik Artis

(Tabel bisa disesuaikan, namun minimal ada ini). Semua persoalan di fitur tambahan **wajib diimplementasikan pada basis data**.

Tujuan	Membuat sebuah tabel Statistik Artis yang berisikan ArtisID, total lagu, total musik video, total video ekstra. Isi tabel tersebut hanya berdasarkan dengan data yang ada pada tabel lagu, video musik, video ekstra saja
Query Testing	<pre> SHOW FULL TABLES LIKE 'statistik_artis'; DESC statistik_artis; SELECT * FROM statistik_artis; /* Cek apakah data sesuai dengan mengambil 1 sampel */ SET @max_id_statistik_artis = (SELECT MAX(artis_id) FROM statistik_artis); SELECT </pre>

```

        artis_id,
        COUNT(id) AS total_lagu
FROM
    lagu
WHERE
    artis_id = @max_id_statistik_artis
GROUP BY
    artis_id;

SELECT
        artis_id,
        COUNT(id) AS total_video_musik
FROM
    video_musik
WHERE
    artis_id = @max_id_statistik_artis
GROUP BY
    artis_id;

SELECT
        artis_id,
        COUNT(id) AS total_video_extra
FROM
    video_extra
WHERE
    artis_id = @max_id_statistik_artis
GROUP BY
    artis_id;

/* Ini untuk mengecek pengguna yang tidak ada di
tabel statistik */

SET @max_id_not_in_statistik_artis = (
    SELECT
        MAX(aid.artis_id)
    FROM
        (
            (
                (
                    SELECT
                        id AS artis_id
                    FROM
                        apple_id
                )
            )
        )
    EXCEPT
    (
        (
            SELECT
                artis_id
            FROM
                statistik_artis
        )
    )
) AS aid
);

SELECT
        artis_id,
        COUNT(id) AS lagu

```

	<pre> FROM lagu WHERE artis_id = @max_id_not_in_statistik_artis GROUP BY artis_id; SELECT artis_id, COUNT(id) AS total_video_musik FROM video_musik WHERE artis_id = @max_id_not_in_statistik_artis GROUP BY artis_id; SELECT artis_id, COUNT(id) AS total_video_extra FROM video_extra WHERE artis_id = @max_id_not_in_statistik_artis GROUP BY artis_id; </pre>
Query Manipulasi	<pre> /* Buat fungsi `count_song` untuk menghitung total lagu dari Artis ID `artis_id` */ DROP FUNCTION IF EXISTS count_song; DELIMITER \$\$ CREATE FUNCTION count_song (artis_id INT) RETURNS INT BEGIN DECLARE total INT; SELECT COUNT(l.id) INTO total FROM lagu AS l WHERE l.artis_id = artis_id GROUP BY l.artis_id; RETURN IFNULL(total, 0); END \$\$ DELIMITER ; </pre> <p style="text-align: right;">/* Buat fungsi `count_music_video` untuk menghitung</p>

```

total video musik dari Artis ID `artis_id` */
DROP FUNCTION IF EXISTS count_music_video;

DELIMITER $$

CREATE FUNCTION count_music_video (
    artis_id INT
)
RETURNS INT
BEGIN
    DECLARE total INT;

    SELECT
        COUNT(vm.id) INTO total
    FROM
        video_musik AS vm
    WHERE
        vm.artis_id = artis_id
    GROUP BY
        vm.artis_id;

    RETURN IFNULL(total, 0);
END $$

DELIMITER ;

/* Buat fungsi `count_extra_video` untuk menghitung
total video extra dari Artis ID `artis_id` */

DROP FUNCTION IF EXISTS count_extra_video;

DELIMITER $$

CREATE FUNCTION count_extra_video (
    artis_id INT
)
RETURNS INT
BEGIN
    DECLARE total INT;

    SELECT
        COUNT(ve.id) INTO total
    FROM
        video_extra AS ve
    WHERE
        ve.artis_id = artis_id
    GROUP BY
        ve.artis_id;

    RETURN IFNULL(total, 0);
END $$

DELIMITER ;

/* Buat tabel `statistik_artis` sesuai tujuan yang

```

```

tertulis */

DROP TABLE IF EXISTS statistik_artis;

CREATE TABLE statistik_artis (
    artis_id
        INT,
    total_lagu
        INT UNSIGNED
        NOT NULL
        DEFAULT 0,
    total_video_musik
        INT UNSIGNED
        NOT NULL
        DEFAULT 0,
    total_video_extra
        INT UNSIGNED
        NOT NULL
        DEFAULT 0,
    PRIMARY KEY (artis_id),
    FOREIGN KEY (artis_id) REFERENCES apple_id (id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    CONSTRAINT valid_artis
        CHECK (
            total_lagu > 0
            OR total_video_musik > 0
            OR total_video_extra > 0
        )
);

/* Seed tabel `statistik_artis` */

INSERT INTO statistik_artis (
    artis_id,
    total_lagu,
    total_video_musik,
    total_video_extra
) (
    SELECT
        aid.artis_id,
        count_song(aid.artis_id),
        count_music_video(aid.artis_id),
        count_extra_video(aid.artis_id)
    FROM
        (
            (
                SELECT DISTINCT
                    artis_id
                FROM
                    lagu
            )
            UNION
            (
                SELECT DISTINCT

```

	<pre> artis_id FROM video_musik) UNION (SELECT DISTINCT artis_id FROM video_extra)) aid); </pre>
Tangkapan Layar Sebelum Manipulasi	
<pre> MariaDB [tubes_if2240_apple_music]> MariaDB [tubes_if2240_apple_music]> SHOW FULL TABLES LIKE 'statistik_artis'; Empty set (0,000 sec) MariaDB [tubes_if2240_apple_music]> MariaDB [tubes_if2240_apple_music]> DESC statistik_artis; ERROR 1146 (42S02): Table 'tubes_if2240_apple_music.statistik_artis' doesn't exist MariaDB [tubes_if2240_apple_music]> MariaDB [tubes_if2240_apple_music]> SELECT * > FROM > statistik_artis; ERROR 1146 (42S02): Table 'tubes_if2240_apple_music.statistik_artis' doesn't exist MariaDB [tubes_if2240_apple_music]> /* Cek apakah data sesuai dengan mengambil 1 sampel */ MariaDB [tubes_if2240_apple_music]> SET @max_id_statistik_artis = (> SELECT > MAX(artis_id) > FROM > statistik_artis >); ERROR 1146 (42S02): Table 'tubes_if2240_apple_music.statistik_artis' doesn't exist MariaDB [tubes_if2240_apple_music]> MariaDB [tubes_if2240_apple_music]> SELECT > artis_id, > COUNT(id) AS total_lagu > FROM > lagu > WHERE > artis_id = @max_id_statistik_artis > GROUP BY > artis_id; Empty set (0,000 sec) MariaDB [tubes_if2240_apple_music]> MariaDB [tubes_if2240_apple_music]> SELECT > artis_id, > COUNT(id) AS total_video_musik > FROM > video_musik > WHERE > artis_id = @max_id_statistik_artis > GROUP BY > artis_id; Empty set (0,000 sec) MariaDB [tubes_if2240_apple_music]> MariaDB [tubes_if2240_apple_music]> SELECT > artis_id, > COUNT(id) AS total_video_extra > FROM > video_extra > WHERE > artis_id = @max_id_statistik_artis > GROUP BY > artis_id; Empty set (0,000 sec) MariaDB [tubes_if2240_apple_music]> MariaDB [tubes_if2240_apple_music]> /* Ini untuk mengecek pengguna yang tidak ada di tabel statistik */ MariaDB [tubes_if2240_apple_music]> MariaDB [tubes_if2240_apple_music]> SET @max_id_not_in_statistik_artis = (> SELECT > MAX(@id.artis_id) > FROM > (> SELECT > id AS artis_id > FROM > apple_id >) cursor </pre>	

```

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Ini untuk mengecek pengguna yang tidak ada di tabel statistik */
MariaDB [tubes_if2240_apple_music]> SET @max_id_not_in_statistik_artis =
--> SELECT
--> MAX(aid.artis_id)
--> FROM
--> (
--> (
--> SELECT
--> id AS artis_id
--> FROM
--> apple_id
--> )
--> )EXCEPT
--> (
--> SELECT
--> artis_id
--> FROM
--> statistik_artis
--> )
--> ) AS aid
--> ;
ERROR 1146 (42S02): Table 'tubes_if2240_apple_music.statistik_artis' doesn't exist
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> SELECT
--> artis_id,
--> COUNT(id) AS lagu
--> FROM
--> lagu
--> WHERE
--> artis_id = @max_id_not_in_statistik_artis
--> GROUP BY
--> artis_id;
Empty set (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> SELECT
--> artis_id,
--> COUNT(id) AS lagu
--> FROM
--> lagu
--> WHERE
--> artis_id = @max_id_not_in_statistik_artis
--> GROUP BY
--> artis_id;
Empty set (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> SELECT
--> artis_id,
--> COUNT(id) AS total_video_musik
--> FROM
--> video_musik
--> WHERE
--> artis_id = @max_id_not_in_statistik_artis
--> GROUP BY
--> artis_id;
Empty set (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> SELECT
--> artis_id,
--> COUNT(id) AS total_video_extra
--> FROM
--> vvideo_extra
--> WHERE
--> artis_id = @max_id_not_in_statistik_artis
--> GROUP BY
--> artis_id;
Empty set (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DROP FUNCTION IF EXISTS count_song;
Query OK, 0 rows affected, 1 warning (0,019 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> -
MariaDB [tubes_if2240_apple_music]> /* Buat fungsi 'count_song' untuk menghitung total lagu dari Artis ID 'artis_id' */
MariaDB [tubes_if2240_apple_music]> MariaDB [tubes_if2240_apple_music]> DROP FUNCTION IF EXISTS count_music_video;
Query OK, 0 rows affected, 1 warning (0,020 sec)

```

Tangkapan Layar Proses Manipulasi

```

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> -
MariaDB [tubes_if2240_apple_music]> /* Buat fungsi 'count_song' untuk menghitung total lagu dari Artis ID 'artis_id' */
MariaDB [tubes_if2240_apple_music]> MariaDB [tubes_if2240_apple_music]> DROP FUNCTION IF EXISTS count_song;
Query OK, 0 rows affected, 1 warning (0,019 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELIMITER $$
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> CREATE FUNCTION count_song (
--> artis_id INT
--> )
--> RETURNS INT
--> BEGIN
--> DECLARE total INT;
-->
--> SELECT
--> COUNT(l.id) INTO total
--> FROM
--> lagu AS l
--> WHERE
--> l.artis_id = artis_id
--> GROUP BY
--> l.artis_id;
-->
--> RETURN IFNULL(total, 0);
--> END $$
Query OK, 0 rows affected (0,284 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELIMITER ;
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Buat fungsi 'count_music_video' untuk menghitung total video musik dari Artis ID 'artis_id' */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DROP FUNCTION IF EXISTS count_music_video;
Query OK, 0 rows affected, 1 warning (0,020 sec)

```

```

MariaDB [tubes_lf2240_apple_music]> DELIMITER ;
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /* Buat fungsi 'count_music_video' untuk menghitung total video musik dari Artis ID 'artis_id' */
MariaDB [tubes_lf2240_apple_music]> DROP FUNCTION IF EXISTS count_music_video;
Query OK, 0 rows affected, 1 warning (0,020 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DELIMITER $$
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> CREATE FUNCTION count_music_video (
    >     artis_id INT
    > )
    > RETURNS INT
    > BEGIN
    >     DECLARE total INT;
    >
    >     SELECT
    >         COUNT(vm.id) INTO total
    >     FROM
    >         video_musik AS vm
    >     WHERE
    >         vm.artis_id = artis_id
    >     GROUP BY
    >         vm.artis_id;
    >
    >     RETURN IFNULL(total, 0);
    > END $$
Query OK, 0 rows affected (0,126 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /* Buat fungsi 'count_extra_video' untuk menghitung total video extra dari Artis ID 'artis_id' */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DROP FUNCTION IF EXISTS count_extra_video;
Query OK, 0 rows affected, 1 warning (0,001 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DELIMITER $$
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> CREATE FUNCTION count_extra_video (
    >     artis_id INT
    > )
    > RETURNS INT
    > BEGIN
    >     DECLARE total INT;
    >
    >     SELECT
    >         COUNT(ve.id) INTO total
    >     FROM
    >         video_extra AS ve
    >     WHERE
    >         ve.artis_id = artis_id
    >     GROUP BY
    >         ve.artis_id;
    >
    >     RETURN IFNULL(total, 0);
    > END $$
Query OK, 0 rows affected (0,101 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DELIMITER ;
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /* Buat tabel 'statistik_artis' sesuai tujuan yang tertulis */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DROP TABLE IF EXISTS statistik_artis;
Query OK, 0 rows affected, 1 warning (0,115 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /* Buat tabel 'statistik_artis' sesuai tujuan yang tertulis */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DROP TABLE IF EXISTS statistik_artis;
Query OK, 0 rows affected, 1 warning (0,115 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> CREATE TABLE statistik_artis (
    >     artis_id INT,
    >     total_lagu INT UNSIGNED NOT NULL DEFAULT 0,
    >     total_video_musik INT UNSIGNED NOT NULL DEFAULT 0,
    >     total_video_extra INT UNSIGNED NOT NULL DEFAULT 0,
    >     PRIMARY KEY (artis_id),
    >     FOREIGN KEY (artis_id) REFERENCES apple_id (id)
    >     ON UPDATE CASCADE
    >     ON DELETE RESTRICT,
    >     CONSTRAINT valid_artis
    >     CHECK (
        >         total_lagu > 0
        >     OR total_video_musik > 0
        >     OR total_video_extra > 0
        > );
    > );
Query OK, 0 rows affected (0,290 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /* Seed tabel 'statistik_artis' */
MariaDB [tubes_lf2240_apple_music]>

```

```

MariaDB [tubes_lf2240_apple_music]> /* Seed tabel `statistik_artis` */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> INSERT INTO statistik_artis (
-->     artis_id,
-->     total_lagu,
-->     total_video_musik,
-->     total_video_extra
--> ) (
-->     SELECT
-->         aid.artis_id,
-->         count_song(aid.artis_id),
-->         count_music_video(aid.artis_id),
-->         count_extra_video(aid.artis_id)
-->     FROM
-->     (
-->         SELECT DISTINCT
-->             artis_id
-->         FROM
-->             lagu
-->     )
-->     UNION
-->     (
-->         SELECT DISTINCT
-->             artis_id
-->         FROM
-->             video_musik
-->     )
-->     UNION
-->     (
-->         SELECT DISTINCT
-->             artis_id
-->         FROM
-->             video_extra
-->     )
-->     aid
--> );
Query OK, 91 rows affected (0,425 sec)

```

Tangkapan Layar Setelah Manipulasi

```

MariaDB [tubes_lf2240_apple_music]> SHOW FULL TABLES LIKE 'statistik_artis';
+-----+-----+
| Tables_in_tubes_lf2240_apple_music | statistik_artis | Table_type |
+-----+-----+
| statistik_artis                   |                | BASE TABLE |
+-----+-----+
1 row in set (0,000 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DESC statistik.artis;
+-----+-----+-----+-----+-----+-----+
| Field          | Type       | Null | Key | Default | Extra   |
+-----+-----+-----+-----+-----+-----+
| artis_id       | int(11)    | NO  | PRI | NULL    |          |
| total_lagu     | int(10) unsigned | NO  |     | 0        |          |
| total_video_musik | int(10) unsigned | NO  |     | 0        |          |
| total_video_extra | int(10) unsigned | NO  |     | 0        |          |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0,001 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> SELECT
--> *
--> FROM
--> statistik.artis;
+-----+-----+-----+
| artis_id | total_lagu | total_video_musik | total_video_extra |
+-----+-----+-----+
|      1   |      1     |            0      |        1          |
|      2   |      4     |            3      |        1          |
|      4   |      0     |            0      |        2          |
|      5   |      1     |            1      |        1          |
|      6   |      1     |            2      |        0          |
|      7   |      0     |            0      |        1          |
|      8   |      1     |            0      |        1          |
|      9   |      1     |            1      |        1          |
|     10   |      1     |            0      |        0          |
|     11   |      6     |            7      |        4          |
|     12   |      0     |            0      |        1          |
|     13   |      4     |            5      |        3          |
|     14   |      3     |            3      |        1          |
|     15   |      1     |            2      |        4          |
|     17   |      1     |            2      |        1          |
|     18   |      1     |            0      |        1          |
|     19   |      3     |            2      |        0          |
|     20   |      1     |            1      |        1          |
|     21   |      1     |            1      |        1          |
|     22   |      0     |            0      |        1          |
|     23   |      2     |            3      |        0          |
|     25   |      0     |            0      |        2          |
|     26   |      3     |            3      |        0          |
|     27   |      1     |            0      |        3          |
|     28   |      0     |            0      |        2          |
|     29   |      1     |            0      |        0          |
|     30   |      3     |            3      |        2          |
|     31   |      2     |            2      |        1          |
|     32   |      2     |            2      |        2          |
|     33   |      1     |            2      |        1          |
|     34   |      2     |            0      |        1          |
+-----+-----+-----+
41 rows in set (0,001 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> SELECT
--> *
--> FROM
--> statistik.artis;
+-----+-----+-----+
| artis_id | total_lagu | total_video_musik | total_video_extra |
+-----+-----+-----+
|      1   |      1     |            0      |        1          |
|      2   |      4     |            3      |        1          |
|      4   |      0     |            0      |        2          |
|      5   |      1     |            1      |        1          |
|      6   |      1     |            2      |        0          |
|      7   |      0     |            0      |        1          |
|      8   |      1     |            0      |        1          |
|      9   |      1     |            1      |        1          |
|     10   |      1     |            0      |        0          |
|     11   |      6     |            7      |        4          |
|     12   |      0     |            0      |        1          |
|     13   |      4     |            5      |        3          |
|     14   |      3     |            3      |        1          |
|     15   |      1     |            2      |        4          |
|     17   |      1     |            2      |        1          |
|     18   |      1     |            0      |        1          |
|     19   |      3     |            2      |        0          |
|     20   |      1     |            1      |        1          |
|     21   |      1     |            1      |        1          |
|     22   |      0     |            0      |        1          |
|     23   |      2     |            3      |        0          |
|     25   |      0     |            0      |        2          |
|     26   |      3     |            3      |        0          |
|     27   |      1     |            0      |        3          |
|     28   |      0     |            0      |        2          |
|     29   |      1     |            0      |        0          |
|     30   |      3     |            3      |        2          |
|     31   |      2     |            2      |        1          |
|     32   |      2     |            2      |        2          |
|     33   |      1     |            2      |        1          |
|     34   |      2     |            0      |        1          |
+-----+-----+-----+
41 rows in set (0,001 sec)

```

```

| 77 |   3 |       3 |       1 |
| 78 |   1 |       2 |       0 |
| 79 |   4 |       6 |       1 |
| 80 |   1 |       2 |       0 |
| 81 |   0 |       0 |       1 |
| 82 |   3 |       5 |       1 |
| 83 |   2 |       1 |       2 |
| 84 |   1 |       2 |       4 |
| 85 |   1 |       2 |       1 |
| 86 |   0 |       0 |       1 |
| 87 |   2 |       1 |       1 |
| 88 |   1 |       1 |       0 |
| 89 |   3 |       4 |       1 |
| 90 |   0 |       0 |       3 |
| 91 |   2 |       3 |       2 |
| 92 |   3 |       2 |       2 |
| 93 |   1 |       0 |       0 |
| 94 |   0 |       0 |       3 |
| 95 |   2 |       2 |       0 |
| 96 |   2 |       0 |       0 |
| 97 |   2 |       2 |       1 |
| 98 |   3 |       2 |       1 |
| 100|   2 |       2 |       0 |
+----+----+----+----+
91 rows in set (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Cek apakah data sesuai dengan mengambil 1 sampel */
MariaDB [tubes_if2240_apple_music]> MariaDB [tubes_if2240_apple_music]> SET @max_id_statistik_artis = (
->     SELECT
->         MAX(artist_id)
->     FROM
->         statistik_artis
-> );
Query OK, 0 rows affected (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Cek apakah data sesuai dengan mengambil 1 sampel */
MariaDB [tubes_if2240_apple_music]> MariaDB [tubes_if2240_apple_music]> SET @max_id_statistik_artis = (
->     SELECT
->         MAX(artist_id)
->     FROM
->         statistik_artis
-> );
Query OK, 0 rows affected (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> SELECT
->     artist_id,
->     COUNT(id) AS total_lagu
->   FROM
->     lagu
->   WHERE
->     artist_id = @max_id_statistik_artis
->   GROUP BY
->     artist_id;
+-----+-----+
| artist_id | total_lagu |
+-----+-----+
|      100 |        2 |
+-----+-----+
1 row in set (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> SELECT
->     artist_id,
->     COUNT(id) AS total_video_musik
->   FROM
->     video_musik
->   WHERE
->     artist_id = @max_id_statistik_artis
->   GROUP BY
->     artist_id;
+-----+-----+
| artist_id | total_video_musik |
+-----+-----+
|      100 |        2 |
+-----+-----+
1 row in set (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> SELECT
->     artist_id,
->     COUNT(id) AS total_video_extra
->   FROM
->     video_extra
->   WHERE
->     artist_id = @max_id_statistik_artis
->   GROUP BY
->     artist_id;
Empty set (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Ini untuk mengecek pengguna yang tidak ada di tabel statistik */
MariaDB [tubes_if2240_apple_music]> MariaDB [tubes_if2240_apple_music]> SET @max_id_not_in_statistik_artis = (
->     SELECT
->         MAX(@id.artist_id)
->     FROM
->         artist

```

```

-> artis_id;
Empty set (0,000 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /* Ini untuk mengecek pengguna yang tidak ada di tabel statistik */
MariaDB [tubes_lf2240_apple_music]> SET @max_id_not_in_statistik_artis = (
-> SELECT
-> MAX(@id.artis_id)
-> FROM
-> (
-> (
-> SELECT
-> id AS artis_id
-> FROM
-> apple_id
-> EXCEPT
-> (
-> SELECT
-> artis_id
-> FROM
-> statistik_artis
-> )
-> ) AS ald
-> );
Query OK, 0 rows affected (0,001 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> SELECT
-> artis_id,
-> COUNT(id) AS lagu
-> FROM
-> lagu
-> WHERE
-> artis_id = @max_id_not_in_statistik_artis
-> GROUP BY
-> artis_id;
Empty set (0,000 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> SELECT
-> artis_id,
-> COUNT(id) AS lagu
-> FROM
-> lagu
-> WHERE
-> artis_id = @max_id_not_in_statistik_artis
-> GROUP BY
-> artis_id;
Empty set (0,000 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> SELECT
-> artis_id,
-> COUNT(id) AS total_video_musik
-> FROM
-> video_musik
-> WHERE
-> artis_id = @max_id_not_in_statistik_artis
-> GROUP BY
-> artis_id;
Empty set (0,000 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> SELECT
-> artis_id,
-> COUNT(id) AS total_video_extra
-> FROM
-> vvideo_extra
-> WHERE
-> artis_id = @max_id_not_in_statistik_artis
-> GROUP BY
-> artis_id;
Empty set (0,000 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> ■

```

d. Update Statistik Artis

(Tabel bisa disesuaikan, namun minimal ada ini). Semua persoalan di fitur tambahan **wajib diimplementasikan pada basis data**.

Tujuan	Membuat skema untuk mengupdate tabel Statistik Artis ketika data terdapat <i>insertion</i> pada tabel Lagu, Video Musik, dan Video Ekstra
Query Testing	<pre> SELECT * FROM statistik_artis </pre>

	<pre> WHERE artis_id IN (1, 3, 16); </pre>
Query Manipulasi	<pre> /* Membuat stored function `is_artis_exist_in_statistics` untuk mengecek apakah Artis ID `artis_id` telah ada pada tabel `statistik_artis` */ DROP FUNCTION IF EXISTS is_artis_exist_in_statistics; DELIMITER \$\$ CREATE FUNCTION is_artis_exist_in_statistics (artis_id INT) RETURNS BOOLEAN BEGIN DECLARE is_exist BOOLEAN; SELECT EXISTS (SELECT sa.artis_id FROM statistik_artis AS sa WHERE sa.artis_id = artis_id) INTO is_exist; RETURN is_exist; END \$\$ DELIMITER ; /* Membuat insertion event trigger pada tabel `lagu` */ DROP TRIGGER IF EXISTS after_insert_lagu; DELIMITER \$\$ CREATE TRIGGER after_insert_lagu AFTER INSERT ON lagu FOR EACH ROW BEGIN IF (is_artis_exist_in_statistics(NEW.artis_id)) THEN UPDATE statistik_artis SET total_lagu = total_lagu + 1 WHERE artis_id = NEW.artis_id; ELSE INSERT INTO statistik_artis (artis_id, </pre>

```

        total_lagu
    )
VALUES (
    NEW.artis_id,
    1
)
;
END IF;
END $$

DELIMITER ;

/* Membuat insertion event trigger pada tabel
`video_musik` */

DROP TRIGGER IF EXISTS after_insert_video_musik;

DELIMITER $$

CREATE TRIGGER after_insert_video_musik
AFTER INSERT ON video_musik
FOR EACH ROW
BEGIN
    IF (is_artis_exist_in_statistics(NEW.artis_id))
    THEN
        UPDATE
            statistik_artis
        SET
            total_video_musik =
            total_video_musik + 1
        WHERE
            artis_id = NEW.artis_id;
    ELSE
        INSERT INTO statistik_artis (
            artis_id,
            total_video_musik
        )
        VALUES (
            NEW.artis_id,
            1
        );
    END IF;
END $$

DELIMITER ;

/* Membuat insertion event trigger pada tabel
`video_extra` */

DROP TRIGGER IF EXISTS after_insert_video_extra;

DELIMITER $$

CREATE TRIGGER after_insert_video_extra
AFTER INSERT ON video_extra
FOR EACH ROW
BEGIN

```

```

IF (is_artis_exist_in_statistics(NEW.artis_id))
THEN
    UPDATE
        statistik_artis
    SET
        total_video_extra =
        total_video_extra + 1
    WHERE
        artis_id = NEW.artis_id;
ELSE
    INSERT INTO statistik_artis (
        artis_id,
        total_video_extra
    )
    VALUES (
        NEW.artis_id,
        1
    );
END IF;
END $$

DELIMITER ;

/* Seed `lagu` */

INSERT INTO lagu (
    artis_id,
    label_id,
    judul,
    durasi,
    tanggal_rilis
) VALUES (
    1,
    1,
    'Purple Whispers',
    FLOOR(100 + (240 - 100) * RAND()),
    DATE(NOW())
), (
    1,
    1,
    'Touch of Nightstorms',
    FLOOR(100 + (240 - 100) * RAND()),
    DATE(NOW())
), (
    3,
    1,
    'Pretty Flamenco',
    FLOOR(100 + (240 - 100) * RAND()),
    DATE(NOW())
), (
    3,
    1,
    'Out of Morning Sun',
    FLOOR(100 + (240 - 100) * RAND()),
    DATE(NOW())
), (

```

```

16,
1,
'Running Fire',
FLOOR(100 + (240 - 100) * RAND()), 
DATE(NOW())
), (
16,
1,
'Higher Things',
FLOOR(100 + (240 - 100) * RAND()), 
DATE(NOW())
);

/* Seed `video_musik` */

INSERT INTO video_musik (
lagu_id,
artis_id,
label_id,
judul,
durasi,
tanggal_rilis
) VALUES (
165,
1,
1,
'Purple Whispers MV',
FLOOR(120 + (300 - 120) * RAND()), 
DATE(NOW())
), (
166,
1,
1,
'Touch of Nightstorms MV',
FLOOR(120 + (300 - 120) * RAND()), 
DATE(NOW())
), (
167,
3,
1,
'Pretty Flamenco MV',
FLOOR(120 + (300 - 120) * RAND()), 
DATE(NOW())
), (
168,
3,
1,
'Out of Morning Sun MV',
FLOOR(120 + (300 - 120) * RAND()), 
DATE(NOW())
), (
169,
16,
1,
'Running Fire MV',
FLOOR(120 + (300 - 120) * RAND()), 
DATE(NOW())
)

```

```

        DATE(NOW())
), (
    170,
    16,
    1,
    'Higher Things MV',
    FLOOR(120 + (300 - 120) * RAND()),
    DATE(NOW())
);

/* Seed `video_extra` */

INSERT INTO video_extra (
    artis_id,
    label_id,
    judul,
    durasi,
    tanggal_rilis
) VALUES (
    1,
    1,
    'Purple Whispers VE',
    FLOOR(300 + (900 - 300) * RAND()),
    DATE(NOW())
), (
    1,
    1,
    'Touch of Nightstorms VE',
    FLOOR(300 + (900 - 300) * RAND()),
    DATE(NOW())
), (
    3,
    1,
    'Pretty Flamenco VE',
    FLOOR(300 + (900 - 300) * RAND()),
    DATE(NOW())
), (
    3,
    1,
    'Out of Morning Sun VE',
    FLOOR(300 + (900 - 300) * RAND()),
    DATE(NOW())
), (
    16,
    1,
    'Running Fire VE',
    FLOOR(300 + (900 - 300) * RAND()),
    DATE(NOW())
), (
    16,
    1,
    'Higher Things VE',
    FLOOR(300 + (900 - 300) * RAND()),
    DATE(NOW())
);

```

Tangkapan Layar Sebelum Manipulasi

```
MariaDB [tubes_lf2240_apple_music]> SELECT
--> *
--> FROM
--> statistik_artis
--> WHERE
--> artis_id IN (1, 3, 16);
+-----+
| artis_id | total_lagu | total_video_musik | total_video_extra |
+-----+
| 1 | 1 | 0 | 1 |
+-----+
1 row in set (0,000 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]>
```

Tangkapan Layar Proses Manipulasi

```
MariaDB [tubes_lf2240_apple_music]> /* Membuat stored function 'is_artis_exist_in_statistics' untuk mengecek apakah Artis ID 'artis_id' telah ada pada
tabel 'statistik_artis' */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DROP FUNCTION IF EXISTS is_artis_exist_in_statistics;
Query OK, 0 rows affected, 1 warning (0,034 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DELIMITER $$

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> CREATE FUNCTION is_artis_exist_in_statistics (
--> artis_id INT
--> )
--> RETURNS BOOLEAN
--> BEGIN
--> DECLARE is_exist BOOLEAN;
-->
--> SELECT
--> EXISTS (
--> SELECT
--> sa.artis_id
--> FROM
--> statistik_artis AS sa
--> WHERE
--> sa.artis_id = artis_id
--> ) INTO is_exist;
-->
--> RETURN is_exist;
--> END $$
Query OK, 0 rows affected (0,169 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DELIMITER ;
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /* Membuat insertion event trigger pada tabel 'lagu' */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> DROP TRIGGER IF EXISTS after_insert_lagu;
Query OK, 0 rows affected, 1 warning (0,000 sec)
```

```

MariaDB [tubes_if2240_apple_music]> DELIMITER ;
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Membuat insertion event trigger pada tabel 'lagu' */
MariaDB [tubes_if2240_apple_music]> DROP TRIGGER IF EXISTS after_insert_lagu;
Query OK, 0 rows affected, 1 warning (0,000 sec)

MariaDB [tubes_if2240_apple_music]> DELIMITER $$ 
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> CREATE TRIGGER after_insert_lagu
    --> AFTER INSERT ON lagu
    --> FOR EACH ROW
    --> BEGIN
    -->   IF (is_artis_exist_in_statistics(NEW.artis_id)) THEN
    -->     UPDATE statistik_artis
    -->       SET
    -->         total_lagu = total_lagu + 1
    -->       WHERE
    -->         artis_id = NEW.artis_id;
    -->   ELSE
    -->     INSERT INTO statistik_artis (
    -->       artis_id,
    -->       total_lagu
    -->     )
    -->     VALUES (
    -->       NEW.artis_id,
    -->       1
    -->     );
    -->   END IF;
    --> END $$;
Query OK, 0 rows affected (0,235 sec)

MariaDB [tubes_if2240_apple_music]> DELIMITER ;
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Membuat insertion event trigger pada tabel 'video_musik' */
MariaDB [tubes_if2240_apple_music]> DROP TRIGGER IF EXISTS after_insert_video_musik;
Query OK, 0 rows affected, 1 warning (0,000 sec)

MariaDB [tubes_if2240_apple_music]> DELIMITER $$ 
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> CREATE TRIGGER after_insert_video_musik
    --> AFTER INSERT ON video_musik
    --> FOR EACH ROW
    --> BEGIN
    -->   IF (is_artis_exist_in_statistics(NEW.artis_id)) THEN
    -->     UPDATE statistik_artis
    -->       SET
    -->         total_video_musik = total_video_musik + 1
    -->       WHERE
    -->         artis_id = NEW.artis_id;
    -->   ELSE
    -->     INSERT INTO statistik_artis (
    -->       artis_id,
    -->       total_video_musik
    -->     )
    -->     VALUES (
    -->       NEW.artis_id,
    -->       1
    -->     );
    -->   END IF;
    --> END $$;
Query OK, 0 rows affected (0,324 sec)

MariaDB [tubes_if2240_apple_music]> DELIMITER ;
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Membuat insertion event trigger pada tabel 'video_extra' */
MariaDB [tubes_if2240_apple_music]> DROP TRIGGER IF EXISTS after_insert_video_extra;
Query OK, 0 rows affected, 1 warning (0,000 sec)

MariaDB [tubes_if2240_apple_music]> DELIMITER $$ 
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> CREATE TRIGGER after_insert_video_extra
    --> AFTER INSERT ON video_extra
    --> FOR EACH ROW
    --> BEGIN
    -->   IF (is_artis_exist_in_statistics(NEW.artis_id)) THEN
    -->     UPDATE statistik_artis
    -->       SET
    -->         total_video_extra = total_video_extra + 1
    -->       WHERE
    -->         artis_id = NEW.artis_id;
    -->   ELSE
    -->     INSERT INTO statistik_artis (
    -->       artis_id,
    -->       total_video_extra
    -->     )
    -->     VALUES (
    -->       NEW.artis_id,
    -->       1
    -->     );
    -->   END IF;
    --> END $$;
Query OK, 0 rows affected (0,255 sec)

MariaDB [tubes_if2240_apple_music]> DELIMITER ;
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Seed 'lagu' */
MariaDB [tubes_if2240_apple_music]>

```

```

MariaDB [tubes_if2240_apple_music]> /* Seed `lagu` */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> INSERT INTO lagu (
--> artis_id,
--> label_id,
--> judul,
--> durasi,
--> tanggal_rilis
--> ) VALUES (
--> 1,
--> 1,
--> 'Purple Whispers',
--> FLOOR(100 + (240 - 100) * RAND()),
--> DATE(NOW())
--> ), (
--> 1,
--> 1,
--> 'Touch of Nightstorms',
--> FLOOR(100 + (240 - 100) * RAND()),
--> DATE(NOW())
--> ), (
--> 3,
--> 3,
--> 1,
--> 'Pretty Flamenco',
--> FLOOR(100 + (240 - 100) * RAND()),
--> DATE(NOW())
--> ), (
--> 3,
--> 3,
--> 1,
--> 'Out of Morning Sun',
--> FLOOR(100 + (240 - 100) * RAND()),
--> DATE(NOW())
--> ), (
--> 16,
--> 1,
--> 'Running Fire',
--> FLOOR(100 + (240 - 100) * RAND()),
--> DATE(NOW())
--> ), (
--> 16,
--> 1,
--> 'Running Fire',
--> FLOOR(100 + (240 - 100) * RAND()),
--> DATE(NOW())
--> ), (
--> 16,
--> 1,
--> 'Higher Things',
--> FLOOR(100 + (240 - 100) * RAND()),
--> DATE(NOW())
--> );
Query OK, 6 rows affected (0.064 sec)
Records: 6  Duplicates: 0  Warnings: 0

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Seed `video_musik` */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> INSERT INTO video_musik (
--> lagu_id,
--> artis_id,
--> label_id,
--> judul,
--> durasi,
--> tanggal_rilis
--> ) VALUES (
--> 165,
--> 1,
--> 1,
--> 'Purple Whispers MV',
--> FLOOR(120 + (300 - 120) * RAND()),
--> DATE(NOW())
--> ), (
--> 166,
--> 1,
--> 1,
--> 'Touch of Nightstorms MV',
--> FLOOR(120 + (300 - 120) * RAND()),
--> DATE(NOW())
--> ), (
--> 167,
--> 3,
--> 1,
--> 'Pretty Flamenco MV',
--> FLOOR(120 + (300 - 120) * RAND()),
--> DATE(NOW())
--> ), (
--> 168,
--> 3,
--> 1,
--> 'Out of Morning Sun MV',
--> FLOOR(120 + (300 - 120) * RAND()),
--> DATE(NOW())
--> ), (
--> 169,
--> 16,
--> 1,
--> 'Running Fire MV',
--> FLOOR(120 + (300 - 120) * RAND()),
--> DATE(NOW())
--> ), (
--> 170,
--> 16,
--> 1,
--> 'Higher Things MV',
--> FLOOR(120 + (300 - 120) * RAND()),
--> DATE(NOW())
--> );
Query OK, 10 rows affected (0.064 sec)
Records: 10  Duplicates: 0  Warnings: 0

ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails ('tubes_if2240_apple_music`.`video_musik`, CONSTRAINT `video_musi

```

```

--> );
ERROR 1452 (23000): Cannot add or update a child
k_ibfk_2 FOREIGN KEY (`lagu_id`) REFERENCES `lagu`(`id`) ON UPDATE CASCADE)
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /* Seed `video_extra` */
MariaDB [tubes_lf2240_apple_music]> INSERT INTO video_extra (
--> artis_id,
--> label_id,
--> judul,
--> durasi,
--> tanggal_rilis
--> ) VALUES (
--> 1,
--> 1,
--> 'Purple Whispers VE',
--> FLOOR(300 + (900 - 300) * RAND()),
--> DATE(NOW())
--> ),
--> 1,
--> 1,
--> 'Touch of Nightstorms VE',
--> FLOOR(300 + (900 - 300) * RAND()),
--> DATE(NOW())
--> ),
--> 1,
--> 1,
--> 'Pretty Flamenco VE',
--> FLOOR(300 + (900 - 300) * RAND()),
--> DATE(NOW())
--> ),
--> 1,
--> 1,
--> 'Out of Morning Sun VE',
--> FLOOR(300 + (900 - 300) * RAND()),
--> DATE(NOW())
--> ),
--> 1,
--> 1,
--> 'Running Fire VE',
--> FLOOR(300 + (900 - 300) * RAND()),
--> DATE(NOW())
--> ),
--> 1,
--> 1,
--> 'Higher Things VE',
--> FLOOR(300 + (900 - 300) * RAND()),
--> DATE(NOW())
--> );
Query OK, 6 rows affected (0,019 sec)
Records: 6 Duplicates: 0 Warnings: 0
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> █

```

Tangkapan Layar Setelah Manipulasi (Ekspektasi statistik berubah)

```

MariaDB [tubes_lf2240_apple_music]> SELECT
--> FROM
--> statistik_artis
--> WHERE
--> artis_id IN (1, 3, 16);
+-----+-----+-----+
| artis_id | total_lagu | total_video_musik | total_video_extra |
+-----+-----+-----+
| 1 | 3 | 0 | 3 |
| 3 | 2 | 0 | 2 |
| 16 | 2 | 0 | 2 |
+-----+-----+-----+
3 rows in set (0,000 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> █

```

e. The Only One

(Tabel bisa disesuaikan, namun minimal ada ini). Semua persoalan di fitur tambahan **wajib diimplementasikan pada basis data**.

Tujuan	Membuat validasi pada tabel Konten Produk Komersial sehingga dapat menolak operasi <i>insertion</i> apabila data Produk Komersial pada <i>tuple</i> data tambahan memiliki tipe “Single” dan sudah memiliki lagu didalamnya.
Query Testing	<pre> /* Instance pada database saat ini antara relasi `lagu`, `produk_komersial`, dan `lagu_produk_komersial` adalah setiap `artis` masing-masing hanya memiliki 1 buah `produk_komersial` yang terdiri dari sejumlah `lagu`. Oleh karena itu, perlu dilakukan penyesuaian agar dapat dilakukan testing terhadap persoalan ini. Penyesuaian yang dilakukan adalah dengan melakukan insert sebuah `lagu` seorang `artis` yang lagu tersebut akan dimasukkan ke dalam `produk_komersial` si `artis` tersebut, yang dalam hal ini, `artis` tersebut adalah `artis` yang memiliki `produk_komersial` bertipe `Single` */ /* Simpan salah satu `produk_komersial`.`id` yang bertipe `Single` dan telah berisi lagu, yang dalam hal ini, akan diambil `produk_komersial` yang memenuhi kriteria tersebut dan ID-nya bernilai maksimum. Selain itu, simpan juga `artis`.`id` dari `produk_komersial` yang terpilih tersebut. */ SELECT pk.id, pk.artis_id INTO @max_pk_id_single_type_has_song, @artis_id_of_max_pk_id_single_type_has_song FROM produk_komersial AS pk INNER JOIN lagu_produk_komersial AS lpk ON (pk.id = lpk.produk_komersial_id) WHERE pk.tipe = 'Single' GROUP BY pk.id </pre>

```

        ORDER BY
            pk.id ASC
        LIMIT 1;

        /* Akan dilakukan insert data pada tabel `lagu` untuk
        nantinya dimasukkan ke dalam target
        `produk_komersial` yang sebelumnya telah disimpan.
        Namun, sebelum itu, pastikan dulu bahwa status
        berlangganan dari `produk_komersial`.`artis_id`
        adalah `aktif` sebagai efek dari implementasi trigger
        pada persoalan sebelumnya */

        UPDATE
            subscription
        SET
            status = 'aktif'
        WHERE
            (subscription_id, pengguna_id) IN (
                SELECT
                    *
                FROM (
                    SELECT
                        subscription_id,
                        pengguna_id
                    FROM
                        subscription
                    WHERE
                        pengguna_id =
                            @max_pk_id_single_type_has_s
                            ong
                    ORDER BY
                        subscription_id DESC
                    LIMIT 1
                ) AS s
            );
        /* Akan dilakukan insert `lagu` untuk nantinya
        dimasukkan ke dalam `produk_komersial` yang
        sebelumnya telah disimpan */

        SET @new_max_lagu_id = (
            SELECT
                MAX(id) + 1
            FROM
                lagu
        );

        INSERT INTO lagu (
            id,
            artis_id,
            label_id,
            judul,
            durasi,
            tanggal_rilis
        ) VALUES (
            @new_max_lagu_id,

```

	<pre> @artis_id_of_max_pk_id_single_type_has_song, 100, 'City of Dreams', FLOOR(100 + (240 - 100) * RAND()), DATE(NOW())); /* Bagian inti dari testing. Melakukan <i>insert</i> pada `lagu_produk_komersial` dengan `lagu`.`id` adalah id maksimum dari target lagu yang lagu tersebut saat ini tidak berada di dalam `produk_komersial` (dengan ID bernilai maksimum) yang bertipe `Single` dan telah berisi lagu. Targetnya, lagu tersebut akan dimasukkan ke `produk_komersial` (dengan ID bernilai maksimum) yang bertipe `Single` dan telah berisi lagu. */ </pre>
Query Manipulasi	<pre> INSERT INTO lagu_produk_komersial (lagu_id, produk_komersial_id) VALUES (@new_max_lagu_id, @max_pk_id_single_type_has_song); /* Clean up data yang sebelumnya di-<i>insert</i> pada tabel `lagu` dan `lagu_produk_komersial` */ DELETE FROM lagu_produk_komersial WHERE lagu_id = @new_max_lagu_id AND produk_komersial_id = @max_pk_id_single_type_has_song; DELETE FROM lagu WHERE id = @new_max_lagu_id; </pre>

```

DELIMITER $$

CREATE FUNCTION
is_produk_komersial_single_and_has_song (
    produk_komersial_id INT
)
RETURNS BOOLEAN
BEGIN
    DECLARE is_exist BOOLEAN;

    SELECT
        EXISTS (
            SELECT
                pk.id
            FROM
                produk_komersial AS pk
                INNER JOIN
                lagu_produk_komersial AS lpk
                    ON (pk.id =
                        lpk.produk_komersial_i
                        d)
            WHERE
                pk.tipe = 'Single'
                AND pk.id =
                produk_komersial_id
        ) INTO is_exist;

    RETURN is_exist;
END $$

DELIMITER ;

/* Buat trigger `before_insert_lagu_produk_komersial` untuk memenuhi tujuan yang tertulis */

DROP TRIGGER IF EXISTS
before_insert_lagu_produk_komersial;

DELIMITER $$

CREATE TRIGGER before_insert_lagu_produk_komersial
BEFORE INSERT ON lagu_produk_komersial
FOR EACH ROW
BEGIN
    IF
        (is_produk_komersial_single_and_has_song(NEW.pr
oduk_komersial_id)) THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT
            = 'Produk komersial bertipe Single dan
            telah berisi lagu';
    END IF;
END $$

DELIMITER ;

```

Tangkapan Layar Sebelum Manipulasi (Ekspektasi proses insertion tetap berjalan)

```
MariaDB [tubes_if2240_apple_music]> SELECT
    *-> FROM
    -> statistik_artis
    -> WHERE
    -> artis_id IN (1, 3, 16);
+-----+-----+-----+
| artis_id | total_lagu | total_video_musik | total_video_extra |
+-----+-----+-----+
| 1 | 3 | 0 | 3 |
| 3 | 2 | 0 | 2 |
| 16 | 2 | 0 | 2 |
+-----+-----+-----+
3 rows in set (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> 
```



```
/*
 * Melakukan insert pada `lagu_produk_komersial` dengan `lagu`.`id` adalah id maksimum dari target lagu yang lagu tersebut saat ini tidak berada di dalam `produk_komersial` (dengan ID bernilai maksimum) yang bertipe 'single' dan telah berisi lagu. Targetnya, lagu tersebut akan dimasukkan ke `produk_komersial` (dengan ID bernilai maksimum) yang bertipe 'Single' dan telah berisi lagu.
 */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> INSERT INTO lagu_produk_komersial (
    -> lagu_id,
    -> produk_komersial_id
    -> )
    -> VALUES (
    -> @new_max_lagu_id,
    -> @max_pk_id_single_type_has_song
    -> );
Query OK, 1 row affected (0,013 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Clean up data yang sebelumnya di-insert pada tabel 'lagu' dan 'lagu_produk_komersial' */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELETE
    -> FROM
    -> lagu_produk_komersial
    -> WHERE
    -> lagu_id = @new_max_lagu_id
    -> AND produk_komersial_id = @max_pk_id_single_type_has_song;
Query OK, 1 row affected (0,050 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELETE
    -> FROM
    -> lagu
    -> WHERE
    -> id = @new_max_lagu_id;
Query OK, 1 row affected (0,010 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> 
```

```

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /*
    >> Instance pada database saat ini antara relasi 'lagu', 'produk_komersial', dan 'lagu_produk_komersial' adalah setiap 'artis' masing-masing hanya memiliki 1 buah 'produk_komersial' yang terdiri dari sejumlah 'lagu'.
    >>
    >> Oleh karena itu, perlu dilakukan penyesuaian agar dapat dilakukan testing terhadap persoalan ini.
    >>
    >> Penyesuaian yang dilakukan adalah dengan melakukan insert sebuah 'lagu' seorang 'artis' yang lagu tersebut akan dimasukkan ke dalam 'produk_komersial' si 'artis' tersebut, yang dalam hal ini, 'artis' tersebut adalah 'artis' yang memiliki 'produk_komersial' bertipe 'single'
    >> */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /* Simpan salah satu 'produk_komersial'. 'id' yang bertipe 'Single' dan telah berisi lagu, yang dalam hal ini, akan diambil produk_komersial yang memenuhi kriteria tersebut dan ID-nya bernilai maksimum. Selain itu, simpan juga 'artis'. 'id' dari 'produk_komersial' yang berpilkil tersebut. */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> SELECT
    >> pk.id,
    >> pk.artis_id,
    >> INTO @max_pk_id_single_type_has_song, @artis_id_of_max_pk_id_single_type_has_song
    >> FROM
    >> produk_komersial AS pk
    >> INNER JOIN lagu_produk_komersial AS lpk
    >> ON (pk.id = lpk.produk_komersial_id)
    >> WHERE
    >> pk.tipe = 'Single'
    >> GROUP BY
    >> pk.id
    >> ORDER BY
    >> pk.id ASC
    >> LIMIT 1;
Query OK, 1 row affected (0,000 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /* Akan dilakukan insert data pada tabel 'lagu' untuk nantinya dimasukkan ke dalam target 'produk_komersial' yang sebelumnya telah disimpan. Namun, sebelum itu, pastikan dulu bahwa status berlangganan dari 'produk_komersial'. 'artis_id' adalah 'aktif' sebagai efek dari implementasi trigger pada persoalan sebelumnya */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> UPDATE
    >> SET
    >> SET
    >> status = 'aktif'
    >> WHERE
    >> (subscription_id, pengguna_id) IN (
    >> SELECT
    >> *
    >> FROM (
    >> SELECT
    >> subscription_id,
    >> pengguna_id
    >> FROM
    >> subscription
    >> WHERE
    >> pengguna_id = @max_pk_id_single_type_has_song
    >> ORDER BY
    >> subscription_id DESC
    >> LIMIT 1
    >> ) AS s
    >> );
Query OK, 0 rows affected (0,001 sec)
Rows matched: 1  Changed: 0  Warnings: 0

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /* Akan dilakukan insert 'lagu' untuk nantinya dimasukkan ke dalam 'produk_komersial' yang sebelumnya telah disimpan */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> SET @new_max_lagu_id = (
    >> SELECT
    >> MAX(id) + 1
    >> );
Query OK, 0 rows affected (0,001 sec)
Rows matched: 1  Changed: 0  Warnings: 0

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /* Akan dilakukan insert 'lagu' untuk nantinya dimasukkan ke dalam 'produk_komersial' yang sebelumnya telah disimpan */
MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> SET @new_max_lagu_id = (
    >> SELECT
    >> MAX(id) + 1
    >> FROM
    >> lagu
    >> );
Query OK, 0 rows affected (0,000 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> INSERT INTO lagu (
    >> id,
    >> artis_id,
    >> label_id,
    >> judul,
    >> durasi,
    >> tanggal_rilis
    >> ) VALUES (
    >> @new_max_lagu_id,
    >> @artis_id_of_max_pk_id_single_type_has_song,
    >> 100,
    >> 'City of Dreams',
    >> FLOOR(100 + (240 - 100) * RAND()),
    >> DATE(NOW())
    >> );
Query OK, 1 row affected (0,025 sec)

MariaDB [tubes_lf2240_apple_music]>
MariaDB [tubes_lf2240_apple_music]> /*
    >> Bagian inti dari testing.
    >> */

```

```

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /*
--> Bagian inti dari testing.
-->
--> Melakukan insert pada 'lagu_produk_komersial' dengan 'lagu'.id adalah id maksimum dari target lagu yang lagu tersebut saat ini tidak berada di dalam 'produk_komersial' (dengan ID bernilai maksimum) yang bertipe 'Single' dan telah berisi lagu. Targetnya, lagu tersebut akan dimasukkan ke 'produk_komersial' (dengan ID bernilai maksimum) yang bertipe 'Single' dan telah berisi lagu.
--> */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> INSERT INTO lagu_produk_komersial (
-->     lagu_id,
-->     produk_komersial_id
--> )
--> VALUES (
-->     @new_max_lagu_id,
-->     @max_pk_id_single_type_has_song
--> );
Query OK, 1 row affected (0,013 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Clean up data yang sebelumnya di-insert pada tabel 'lagu' dan 'lagu_produk_komersial' */
MariaDB [tubes_if2240_apple_music]> DELETE
--> FROM
-->     lagu_produk_komersial
--> WHERE
-->     lagu_id = @new_max_lagu_id
-->     AND produk_komersial_id = @max_pk_id_single_type_has_song;
Query OK, 1 row affected (0,007 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELETE
--> FROM
-->     lagu
--> WHERE
-->     id = @new_max_lagu_id;
-->     AND type = 'Single';

--> Bagian inti dari testing.
-->
--> Melakukan insert pada 'lagu_produk_komersial' dengan 'lagu'.id adalah id maksimum dari target lagu yang lagu tersebut saat ini tidak berada di dalam 'produk_komersial' (dengan ID bernilai maksimum) yang bertipe 'Single' dan telah berisi lagu. Targetnya, lagu tersebut akan dimasukkan ke 'produk_komersial' (dengan ID bernilai maksimum) yang bertipe 'Single' dan telah berisi lagu.
--> */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> INSERT INTO lagu_produk_komersial (
-->     lagu_id,
-->     produk_komersial_id
--> )
--> VALUES (
-->     @new_max_lagu_id,
-->     @max_pk_id_single_type_has_song
--> );
Query OK, 1 row affected (0,013 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Clean up data yang sebelumnya di-insert pada tabel 'lagu' dan 'lagu_produk_komersial' */
MariaDB [tubes_if2240_apple_music]> DELETE
--> FROM
-->     lagu_produk_komersial
--> WHERE
-->     lagu_id = @new_max_lagu_id
-->     AND produk_komersial_id = @max_pk_id_single_type_has_song;
Query OK, 1 row affected (0,007 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELETE
--> FROM
-->     lagu
--> WHERE
-->     id = @new_max_lagu_id;
Query OK, 1 row affected (0,009 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> ■

```

Tangkapan Layar Proses Manipulasi

```

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Buat stored function `is_produk_komersial_single_and_has_song` untuk menentukan apakah produk komersial 'produk_komersial' bertipe 'Single' dan telah berisi lagu */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DROP FUNCTION IF EXISTS is_produk_komersial_single_and_has_song;
Query OK, 0 rows affected, 1 warning (0,029 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELIMITER $$
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> CREATE FUNCTION is_produk_komersial_single_and_has_song (
-->     produk_komersial_id INT
--> )
--> RETURNS BOOLEAN
--> BEGIN
-->     DECLARE is_exist BOOLEAN;
-->
-->     SELECT
-->         EXISTS (
-->             SELECT
-->                 pk.id
-->             FROM
-->                 produk_komersial AS pk
-->                 INNER JOIN lagu_produk_komersial AS lpk
-->                 ON (pk.id = lpk.produk_komersial_id)
-->                 WHERE
-->                     pk.tipe = 'Single'
-->                     AND pk.id = produk_komersial_id
-->             ) INTO is_exist;
-->
-->     RETURN is_exist;
--> END $$
Query OK, 0 rows affected (0,176 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELIMITER ;

```

```

--> produk_komersial AS pk
--> INNER JOIN lagu_produk_komersial AS lpk
--> ON (pk.id = lpk.produk_komersial_id)
--> WHERE
--> pk.tipe = 'Single'
--> AND pk.id = produk_komersial_id
--> ) INTO is_exist;
-->
--> RETURN is_exist;
--> END $$
```

Query OK, 0 rows affected (0,176 sec)

```

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELIMITER ;
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Buat trigger `before_insert_lagu_produk_komersial` untuk memenuhi tujuan yang tertulis */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DROP TRIGGER IF EXISTS before_insert_lagu_produk_komersial;
Query OK, 0 rows affected, 1 warning (0,000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELIMITER $$
```

```

MariaDB [tubes_if2240_apple_music]> CREATE TRIGGER before_insert_lagu_produk_komersial
--> BEFORE INSERT ON lagu_produk_komersial
--> FOR EACH ROW
--> BEGIN
--> IF (is_produk_komersial_single_and_has_song(NEW.produk_komersial.id)) THEN
--> SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Produk Komersial bertipe Single dan telah berisi lagu';
--> END IF;
--> END $$
```

Query OK, 0 rows affected (0,534 sec)

```

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELIMITER ;
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]>
```

Tangkapan Layar Setelah Manipulasi (Ekspektasi akan ditolak oleh trigger)

```

MariaDB [tubes_if2240_apple_music]> /*
--> Instance pada database saat ini antara relasi 'lagu', 'produk_komersial', dan 'lagu_produk_komersial' adalah setiap 'artis' masing-masing hanya memiliki 1 buah 'produk_komersial' yang terdiri dari sejumlah 'lagu'.
--> Oleh karena itu, perlu dilakukan penyesuaian agar dapat dilakukan testing terhadap persoalan ini.
--> */
--> /* Penyesuaian yang dilakukan adalah dengan melakukan insert sebuah 'Lagu' seorang 'artis' yang lagu tersebut akan dimasukkan ke dalam 'produk_komersial' sl 'artis' tersebut, yang dalam hal ini, 'artis' tersebut adalah 'artis' yang memiliki 'produk_komersial' bertipe 'Single'
--> */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Simpan salah satu 'produk_komersial'.id yang bertipe 'Single' dan telah berisi lagu, yang dalam hal ini, akan diambil 'produk_komersial' yang memenuhi kriteria tersebut dan ID-nya bernilai maksimum. Selain itu, simpan juga 'artis'.id dari 'produk_komersial' yang terpilih tersebut. */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> SELECT
-->     pk.id,
-->     pk.artis_id
-->     INTO @max_pk_id_single_type_has_song, @artis_id_of_max_pk_id_single_type_has_song
-->     FROM
-->     produk_komersial AS pk
-->     INNER JOIN lagu_produk_komersial AS lpk
-->     ON (pk.id = lpk.produk_komersial_id)
-->     WHERE
-->     pk.tipe = 'Single'
-->     GROUP BY
-->     pk.id
-->     ORDER BY
-->     pk.id ASC
-->     LIMIT 1;
```

Query OK, 1 row affected (0,001 sec)

```

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Akan dilakukan insert data pada tabel 'lagu' untuk nantinya dimasukkan ke dalam target 'produk_komersial' yang sebelumnya telah disimpan. Namun, sebelum itu, pastikan dulu bahwa status berlangganan dari 'produk_komersial'.artis_id adalah 'aktif' sebagai efek dari implementasi trigger pada persoalan sebelumnya */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> UPDATE
-->     subscription
-->     SET
```

```

MariaDB [tubes_if2240_apple_music]> /* Akan dila... You can paste the image from the clipboard. sukan ke dalam target 'produk_komersial' yang
sebelumnya telah disimpan. Namun, sebelum itu, p... bersial'.'artis_id' adalah 'aktif' sebagai efek
dari implementasi trigger pada persoalan sebelumnya */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> UPDATE
--> subscription
--> SET
--> status = 'aktif'
--> WHERE
--> (subscription_id, pengguna_id) IN (
--> SELECT
--> *
--> FROM (
--> SELECT
--> subscription_id,
--> pengguna_id
--> FROM
--> subscription
--> WHERE
--> pengguna_id = @max_pk_id_single_type_has_song
--> ORDER BY
--> subscription_id DESC
--> LIMIT 1
--> ) AS s
--> );
Query OK, 0 rows affected (0.001 sec)
Rows matched: 1  Changed: 0  Warnings: 0

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Akan dilakukan insert 'lagu' untuk nantinya dimasukkan ke dalam 'produk_komersial' yang sebelumnya telah disimp... */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> SET @new_max_lagu_id = (
--> SELECT
--> MAX(id) + 1
--> FROM
--> lagu
--> );
Query OK, 0 rows affected (0.000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> INSERT INTO lagu (
--> id,
--> artis_id,
--> label_id,
--> judul,
--> durasi,
--> tanggal_rilis
--> ) VALUES (
--> @new_max_lagu_id,
--> @artis_id_of_max_pk_id_single_type_has_song,
--> 100,
--> 'City of Dreams',
--> FLOOR(100 + (240 - 100) * RAND()),
--> DATE(NOW())
--> );
Query OK, 1 row affected (0.046 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /*
--> Bagian inti dari testing.
-->
--> Melakukan insert pada 'lagu_produkt_komersial' dengan 'lagu'.id' adalah id maksimum dari target lagu yang lagu tersebut saat ini tidak berada d... i dalam produk_komersial (dengan ID bernilai maksimum) yang bertipe 'Single' dan telah berisi lagu. Targetnya, lagu tersebut akan dimasukkan ke 'produ...uk_komersial' (dengan ID bernilai maksimum) yang bertipe 'Single' dan telah berisi lagu.
--> */
MariaDB [tubes_if2240_apple_music]> /*
--> Bagian inti dari testing.
-->
--> Melakukan insert pada 'lagu_produkt_komersial' dengan 'lagu'.id' adalah id maksimum dari target lagu yang lagu tersebut saat ini tidak berada d... i dalam produk_komersial (dengan ID bernilai maksimum) yang bertipe 'Single' dan telah berisi lagu. Targetnya, lagu tersebut akan dimasukkan ke 'produ...uk_komersial' (dengan ID bernilai maksimum) yang bertipe 'Single' dan telah berisi lagu.
--> */
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> INSERT INTO lagu_produkt_komersial (
--> lagu_id,
--> produk_komersial_id
--> )
--> VALUES (
--> @new_max_lagu_id,
--> @max_pk_id_single_type_has_song
--> );
ERROR 1644 (45000): Produk komersial bertipe Single dan telah berisi lagu
MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> /* Clean up data yang sebelumnya di-insert pada tabel 'lagu' dan 'lagu_produkt_komersial' */
MariaDB [tubes_if2240_apple_music]> DELETE
--> FROM
--> lagu_produkt_komersial
--> WHERE
--> lagu_id = @new_max_lagu_id
--> AND produk_komersial_id = @max_pk_id_single_type_has_song;
Query OK, 0 rows affected (0.000 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]> DELETE
--> FROM
--> lagu
--> WHERE
--> id = @new_max_lagu_id;
Query OK, 1 row affected (0.008 sec)

MariaDB [tubes_if2240_apple_music]>
MariaDB [tubes_if2240_apple_music]>

```

5. IMPLEMENTASI ACCESS BUSINESS RULE (*jika mengerjakan*)

a. Atribut Turunan Produk Komersial

Dengan konsiderasi bahwa kebutuhan akan total durasi dan juga banyak lagu di suatu produk komersial merupakan sesuatu yang sangat sering untuk di ambil dari basis data, dan mengingat bahwa *query* untuk melakukan hal tersebut jelas tidak “murah” karena memerlukan operasi *count* dan juga *sum* dan juga *join*, ditambah juga skala dari Apple Music yang seharusnya cukup masif, maka alternatif terbaik yang dapat dilakukan adalah melakukan perubahan pada tabel basis data yang ada.

Diperlukan tambahan atribut yaitu *total_durasi* dan juga *banyak_lagu* pada tabel *produk_komersial* dengan tujuan untuk mempercepat akses akan informasi dari kedua hal tersebut. Untuk menjaga *integrity constraint*, akan dibuat sebuah trigger yang tereksekusi setiap kali ada operasi baik *insert, update*, maupun *delete* pada tabel *lagu_produk_komersial*.

Walaupun memang hal ini terkesan menambah *workload* ketika melakukan *operational query*, namun apabila dibandingkan dengan skema sebelumnya yang justru harus melakukan operasi *count*, *sum* dan juga *join* untuk mendapatkan data-data terkait, jelas hal itu menjadi tidak terlalu masalah. Belum lagi, fokus utama dari bisnis Apple Music adalah pada pelanggannya dan bukan terkait dengan proses operasional seperti pemasukan data-data mengenai lagu dan sebagainya. Dapat pula dipilih jam-jam tertentu apabila ingin melakukan proses-proses terkait *operational query* sehingga tidak terlalu mengganggu apa yang dilakukan pengguna.

Strategi ini banyak diimplementasikan oleh perusahaan besar salah satunya adalah Instagram. Instagram menyimpan data jumlah *likes* sesungguhnya dari suatu unggahan, walaupun mereka juga memiliki tabel untuk menyimpan suatu like ini tersosiasi dengan siapa (tabel untuk menyimpan like diberikan oleh siapa, ke postingan mana) yang

sebenarnya juga bisa digunakan untuk mendapatkan total like dengan melakukan operasi *count*.

```
// Ini kode untuk mengubah table sekarang
ALTER TABLE produk_komersial
ADD COLUMN total_songs INT DEFAULT 0,
ADD COLUMN total_duration INT DEFAULT 0;

DROP TRIGGER IF EXISTS after_lagu_produk_komersial_insert;

DELIMITER $$

// Ini kode untuk membuat trigger ketika insert
CREATE TRIGGER after_lagu_produk_komersial_insert
AFTER INSERT ON lagu_produk_komersial
FOR EACH ROW
BEGIN
    UPDATE produk_komersial pk
    JOIN lagu l ON l.id = NEW.lagu_id
    SET pk.total_songs = pk.total_songs + 1,
        pk.total_duration = pk.total_duration + l.duration
    WHERE pk.id = NEW.produk_komersial_id;
END;

DELIMITER ;

DROP TRIGGER IF EXISTS after_lagu_produk_komersial_delete;

DELIMITER $$

// Ini kode untuk membuat trigger ketika delete
CREATE TRIGGER after_lagu_produk_komersial_delete
AFTER DELETE ON lagu_produk_komersial
FOR EACH ROW
BEGIN
    UPDATE produk_komersial pk
    JOIN lagu l ON l.id = OLD.lagu_id
    SET pk.total_songs = pk.total_songs - 1,
        pk.total_duration = pk.total_duration - l.duration
    WHERE pk.id = OLD.produk_komersial_id;
END;

DELIMITER ;

DROP TRIGGER IF EXISTS after_lagu_produk_komersial_update;

DELIMITER $$

// Ini kode untuk membuat trigger ketika update
CREATE TRIGGER after_lagu_produk_komersial_update
AFTER UPDATE ON lagu_produk_komersial
FOR EACH ROW
BEGIN
    IF OLD.lagu_id <> NEW.lagu_id THEN
        UPDATE produk_komersial pk
```

```

        JOIN lagu old_l ON old_l.id = OLD.lagu_id
        JOIN lagu new_l ON new_l.id = NEW.lagu_id
        SET pk.total_duration = pk.total_duration -
old_l.duration + new_l.duration
        WHERE pk.id = OLD.produk_komersial_id;
    END IF;
    IF OLD.produk_komersial_id <> NEW.produk_komersial_id THEN
        UPDATE produk_komersial pk
        JOIN lagu l ON l.id = OLD.lagu_id
        SET pk.total_songs = pk.total_songs - 1,
            pk.total_duration = pk.total_duration - l.duration
        WHERE pk.id = OLD.produk_komersial_id;

        UPDATE produk_komersial pk
        JOIN lagu l ON l.id = NEW.lagu_id
        SET pk.total_songs = pk.total_songs + 1,
            pk.total_duration = pk.total_duration + l.duration
        WHERE pk.id = NEW.produk_komersial_id;
    END IF;
END;

DELIMITER ;

```

b. Library Searching

Beberapa strategi dan teknik yang dapat dilakukan untuk memberikan performa pencarian yang lebih baik relatif terhadap model relasional saat ini adalah dengan mengimplementasikan *materialized view* yang menyimpan `produk_komersial`.`id`, `produk_komersial`.`judul`, dan nama artis dari `produk_komersial` tersebut. Untuk semakin mempercepat proses pencarian, kolom terkait `produk_komersial`.`judul` dan nama artis dapat diberlakukan *indexing*. Pada kasus ini, akan dibuat FULLTEXT index antara `produk_komersial`.`judul` dan nama artis sehingga mengoptimasi *natural language search terms*.

Hanya saja, terkait *materialized view*, perlu diperhatikan bahwa *materialized view* relatif terbatas pada *platform* tertentu, atau dengan kata lain, tidak semua DBMS dapat mengimplementasikan *materialized view* tersebut. Selain itu, data yang tersimpan pada *materialized view* juga memungkinkan tidak *up-to-date* 100% secara *realtime* .

```
/* PostgreSQL */
```

```

CREATE MATERIALIZED VIEW CONCURRENTLY library_searching_mv AS (
    SELECT
        pk.id AS produk_komersial_id,
        pk.judul AS produk_komersial_judul,
        aid.nama_pengguna AS produk_komersial_artis,
        to_tsvector('english', coalesce(pk.judul, '') || ''
        || coalesce(aid.nama_pengguna, '')) AS document_en,
        to_tsvector('indonesian', coalesce(pk.judul, '') || ''
        || coalesce(aid.nama_pengguna, '')) AS document_id,
    FROM
        produk_komersial AS pk
        INNER JOIN apple_id AS aid
        ON (pk.artis_id = aid.id);
);

CREATE INDEX idx_library_searching_mv_document_en ON
library_searching_mv USING GIN (document_en);
CREATE INDEX idx_library_searching_mv_document_id ON
library_searching_mv USING GIN (document_id);

```

c. Apple Music Classical

Salah satu strategi yang dapat dilakukan untuk meningkatkan performa aplikasi Apple Music Classical adalah dengan melakukan duplikasi beberapa tabel, misalnya tabel `lagu`, `produk_komersial`, `lagu_produkt_komersial`, `playlist`, dan `isi_playlist`. Masing-masing tabel hasil duplikasi tersebut hanya berisi data yang berkaitan dengan Apple Music Classical.

Tabel `produk_komersial` perlu diduplikasi agar Apple Music Classical tidak perlu melakukan pencarian dengan melakukan *matching* terhadap `genre` yang bernilai `Classical`. Sebab, kolom `genre` tersebut tidaklah *ter-index* sehingga pencarian terhadap tabel `produk_komersial` utama dengan melakukan *matching* terhadap `genre` tersebut tidaklah efisien. Selain itu, tabel `produk_komersial` dan tabel-tabel lainnya juga relatif perlu diduplikasi dikarenakan relatif paling banyak dilakukan *query* sehingga diperlukannya proses pencarian yang lebih cepat. Masalah tersebut memungkinkan teratas oleh strategi penduplikasian ini akibat sedikitnya data yang perlu dicari pada tabel-tabel hasil duplikasi. Misalnya, pada tabel `produk_komersial`, oleh karena setiap `lagu` memungkinkan memiliki lebih dari satu `produk_komersial` maka data

pada tabel `produk_komersial` utama dapat berjumlah sangat banyak. Pun, kasusnya sangat mirip dengan tabel `lagu_produk_komersial` yang memiliki sangat banyak data. Pada tabel `playlist` dan `isi_playlist`, untuk mengetahui apakah lagu-lagu berikut produk komersialnya termasuk ke dalam `genre` bertipe `Classical` juga perlu dilakukan *join* sejumlah tabel yang tentunya merupakan operasi yang mahal, terlebih jika data masing-masing tabel tersebut sangat banyak. Belum lagi, selain *join operation*, nantinya perlu dilakukan *matching* terhadap `genre` dari `produk_komersial` suatu `lagu` untuk menentukan apakah lagu tersebut merupakan lagu klasik atau bukan.

Walaupun strategi tersebut memungkinkan meningkatkan performa aplikasi Apple Music Classical, dalam hal ini khususnya performa pencarian, namun terdapat sejumlah dampak buruk yang mungkin terjadi pada aplikasi *original* Apple Music. Misalnya, permasalahan terkait konsistensi data antara tabel *original* dan duplikasi tabel tersebut secara *realtime*. Untuk mengatasinya, dapat didefinisikan *trigger* untuk setiap *events* modifikasi (*inserting*, *updating*, dan *deleting*) pada setiap tabel *original* dari tabel duplikasi yang berkaitan dengan modifikasi terkait `lagu` dan `produk_komersial` ber-genre `Classical`. Sayangnya, pengeksekusian *trigger* tersebut tentu menimbulkan *overhead* dalam memodifikasi tabel *original* sehingga proses memodifikasi yang tetap membuat *database* terjaga konsistensinya membutuhkan waktu yang lebih lama. Melalui pendekatan ini, setiap tabel duplikasi tidak boleh dilakukan modifikasi data secara langsung, hanya boleh termodifikasi akibat tereksekusinya *trigger*. Selain permasalahan konsistensi tersebut, dampak buruk yang mungkin dihasilkan adalah besarnya *storage* yang terpakai sebagai akibat dari penduplikasian tabel-tabel tersebut.

Selain strategi penduplikasian tabel *original* menjadi tabel duplikasi terkait Apple Music Classical, strategi lainnya adalah menduplikasi data terkait

Apple Music Classical dengan menyimpannya ke dalam *materialized view*. Pada dasarnya, efek yang dihasilkan dengan menyimpan data hasil duplikasi ke dalam *materialized view* akan mirip-mirip dengan menyimpan data hasil duplikasi ke tabel reguler yang telah disampaikan. Namun, dengan memanfaatkan *materialized view*, DBMS lah yang akan menjaga konsistensi data antara tabel *original* dan *materialized view* sehingga beban tersebut tidak menjadi tanggung jawab *programmer*. Selain itu, sistem *caching* yang umumnya berlaku pada *materialized view* juga dapat meningkatkan performa. Sayangnya, *materialized view* tidak tersedia di beberapa DBMS, misalnya MariaDB, sehingga *materialized view* cukup bergantung pada *platform*. Belum lagi jika mempertimbangkan faktor-faktor seperti kebutuhan untuk migrasi dan lain-lain.

Selain dua strategi sebelumnya, dapat juga dilakukan *properly indexing*, misalnya pada `produk_komersial`.`genre` sehingga pencarian pada tabel *original* untuk Apple Music Classical memungkinkan dilakukan secara lebih cepat. Walaupun begitu, kemungkinan sangat banyaknya data dari tabel-tabel *original* yang telah disebutkan memungkinkan pemberlakuan *indexing* semata tersebut tidak berdampak signifikan terhadap peningkatan performa. Belum lagi, akibat *indexing* tersebut memungkinkan proses modifikasi tabel `produk_komersial` membutuhkan waktu yang lebih lama.

Berikut contoh-contoh *query* implementasinya.

```
/* MariaDB */

/* Menginisialisasi dan seeding tabel `lagu_c` yang berkaitan
dengan genre `klasik` */

CREATE TABLE lagu_c
    id INT AUTO_INCREMENT,
    artis_id INT NOT NULL,
    label_id INT NOT NULL,
    judul VARCHAR(255) NOT NULL,
    durasi INT NOT NULL,
    tanggal_rilis DATE NOT NULL,
```

```

        PRIMARY KEY (id),
        FOREIGN KEY (artis_id) REFERENCES apple_id (id)
            ON UPDATE CASCADE
            ON DELETE RESTRICT,
        FOREIGN KEY (label_id) REFERENCES label (id)
            ON UPDATE CASCADE
            ON DELETE RESTRICT
    );

INSERT INTO lagu_c (
    id,
    artis_id,
    label_id,
    judul,
    durasi,
    tanggal_rilis
) (
    SELECT DISTINCT
        l.id,
        l.artis_id,
        l.label_id,
        l.judul,
        l.durasi,
        l.tanggal_rilis
    FROM
        lagu AS l
    INNER JOIN lagu_produk_komersial AS lpk
        ON (l.id = lpk.lagu_id)
    INNER JOIN produk_komersial AS pk
        ON (lpk.produk_komersial_id = pk.id)
    WHERE
        pk.genre = 'Classical'
);

```

```

/* MariaDB */

/* Menginisialisasi dan seeding tabel `produk_komersial_c` yang
berkaitan dengan genre `klasik` */

CREATE TABLE produk_komersial_c (
    id INT AUTO_INCREMENT,
    artis_id INT NOT NULL,
    judul VARCHAR(255) NOT NULL,
    tipe ENUM('EP','Album','Single'),
    genre VARCHAR(255) NOT NULL,
    tanggal_rilis DATE NOT NULL,

    PRIMARY KEY (id),
    FOREIGN KEY (artis_id) REFERENCES apple_id (id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

```

```

INSERT INTO produk_komersial_c (
    id,
    artis_id,
    judul,
    tipe,
    genre,
    tanggal_rilis
) (
    SELECT
        id,
        artis_id,
        judul,
        tipe,
        genre,
        tanggal_rilis
    FROM
        produk_komersial
    WHERE
        genre = 'Classical'
);

```

```

/* MariaDB */

/* Menginisialisasi dan seeding tabel `lagu_produk_komersial_c` yang berkaitan dengan genre `klasik` */

CREATE TABLE lagu_produk_komersial_c (
    lagu_id INT,
    produk_komersial_id INT NOT NULL,
    PRIMARY KEY (lagu_id, produk_komersial_id),
    FOREIGN KEY (lagu_id) REFERENCES lagu (id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    FOREIGN KEY (produk_komersial_id) REFERENCES
    produk_komersial (id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

INSERT INTO lagu_produk_komersial_c (
    lagu_id,
    produk_komersial_id,
) (
    SELECT
        l.id,
        pk.id
    FROM
        lagu AS l
        INNER JOIN lagu_produk_komersial AS lpk
            ON (l.id = lpk.lagu_id)
        INNER JOIN produk_komersial AS pk
            ON (lpk.produk_komersial_id = pk.id)
    WHERE

```

```
        pk.genre = 'Classical'  
);
```

```
/* MariaDB */  
  
/* Membuat stored function `is_produk_komersial_classical_genre`  
yang akan menentukan apakah `genre` dari `produk_komersial`.`id`  
bernilai `Classical` */  
  
DROP FUNCTION IF EXISTS is_produk_komersial_classical_genre;  
  
DELIMITER $$  
  
CREATE FUNCTION is_produk_komersial_classical_genre (  
    produk_komersial_id  
)  
RETURNS BOOLEAN  
BEGIN  
    DECLARE is_classical_genre BOOLEAN;  
  
    SELECT  
        EXISTS (  
            SELECT  
                id  
            FROM  
                produk_komersial AS pk  
            WHERE  
                pk.id = produk_komersial_id  
                AND genre = 'Classical'  
        )  
        INTO is_classical_genre;  
  
    RETURN is_classical_genre;  
END $$  
  
DELIMITER ;  
  
/* Membuat trigger untuk insertion event pada tabel  
'lagu_produk_komersial` yang akan turut menambahkan data yang ke  
tabel `lagu_c`, 'produk_komersial_c', dan  
'lagu_produk_komersial_c' jika `genre` dari pasangan `lagu` dan  
'produk_komersial` tersebut bernilai `Classical` */  
  
DROP TRIGGER IF EXISTS after_insert_lagu_produk_komersial;  
  
DELIMITER $$  
  
CREATE TRIGGER after_insert_lagu_produk_komersial  
AFTER INSERT ON lagu_produk_komersial  
FOR EACH ROW  
BEGIN  
    IF  
        (is_produk_komersial_classical_genre(NEW.produk_komersial_i  
d)) THEN
```

```

    INSERT INTO lagu_c (
        id,
        artis_id,
        label_id,
        judul,
        durasi,
        tanggal_rilis
    ) (
        SELECT
            id,
            artis_id,
            label_id,
            judul,
            durasi,
            tanggal_rilis
        FROM
            lagu
        WHERE
            id = NEW.lagu_id
    );

```

```

    INSERT INTO produk_komersial_c (
        id,
        artis_id,
        judul,
        tipe,
        genre,
        tanggal_rilis
    ) (
        SELECT
            id,
            artis_id,
            judul,
            tipe,
            genre,
            tanggal_rilis
        FROM
            produk_komersial
        WHERE
            id = NEW.produk_komersial_id
    );

```

```

    INSERT INTO lagu_produk_komersial_c (
        lagu_id,
        produk_komersial_id,
    )
    VALUES (
        NEW.lagu_id,
        NEW.produk_komersial_id
    );
ENDIF;
END $$

DELIMITER ;

```

```

/* PostgreSQL */

/* Membuat materialized view `lagu_c_mv`, `produk_komersial_c_mv`, dan `lagu_produk_komersial_c_mv`. Seluruhnya berkaitan dengan genre `klasik` yang akan ter-update secara berkala setiap kali masing-masing tabel dimodifikasi. Refresh berkala pada materialized view tersebut tidak bersifat blocking sehingga melakukan SELECT pada saat refresh operation sedang berlangsung memungkinkan data pada materialized view belum ter-update (tidak up-to-date secara realtime 100%) */

CREATE MATERIALIZED VIEW CONCURRENTLY lagu_c_mv AS (
    SELECT DISTINCT
        l.id,
        l.artis_id,
        l.label_id,
        l.judul,
        l.durasi,
        l.tanggal_rilis
    FROM
        lagu AS l
        INNER JOIN lagu_produk_komersial AS lpk
            ON (l.id = lpk.lagu_id)
        INNER JOIN produk_komersial AS pk
            ON (lpk.produk_komersial_id = pk.id)
    WHERE
        pk.genre = 'Classical'
);

CREATE MATERIALIZED VIEW CONCURRENTLY produk_komersial_c_mv AS (
    SELECT
        id,
        artis_id,
        judul,
        tipe,
        genre,
        tanggal_rilis
    FROM
        produk_komersial
    WHERE
        genre = 'Classical'
);

CREATE MATERIALIZED VIEW CONCURRENTLY lagu_produk_komersial_c_mv
AS (
    SELECT
        l.id,
        pk.id
    FROM
        lagu AS l
        INNER JOIN lagu_produk_komersial AS lpk
            ON (l.id = lpk.lagu_id)
        INNER JOIN produk_komersial AS pk
            ON (lpk.produk_komersial_id = pk.id)
);

```

```
    WHERE
        pk.genre = 'Classical'
);
```