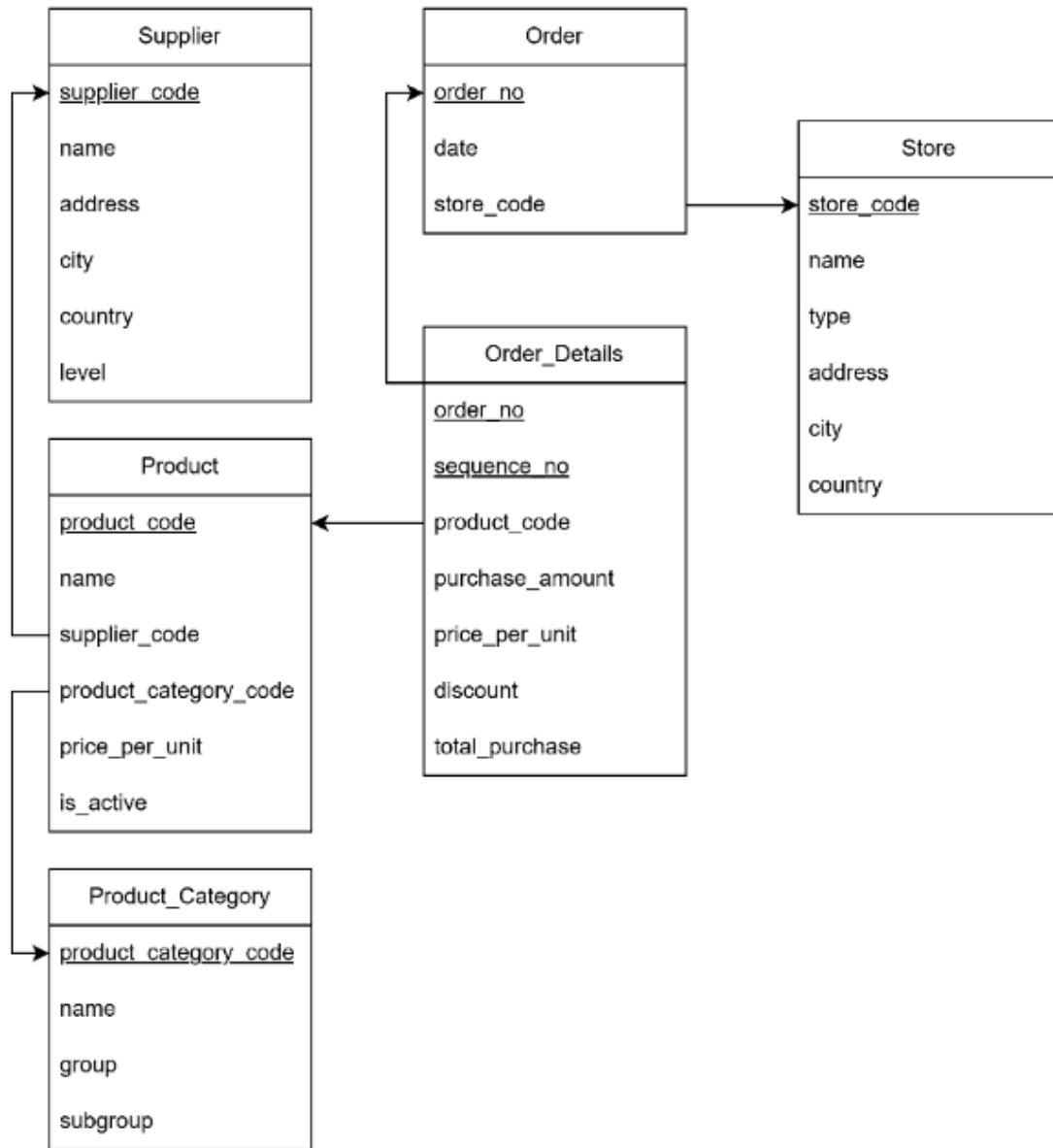
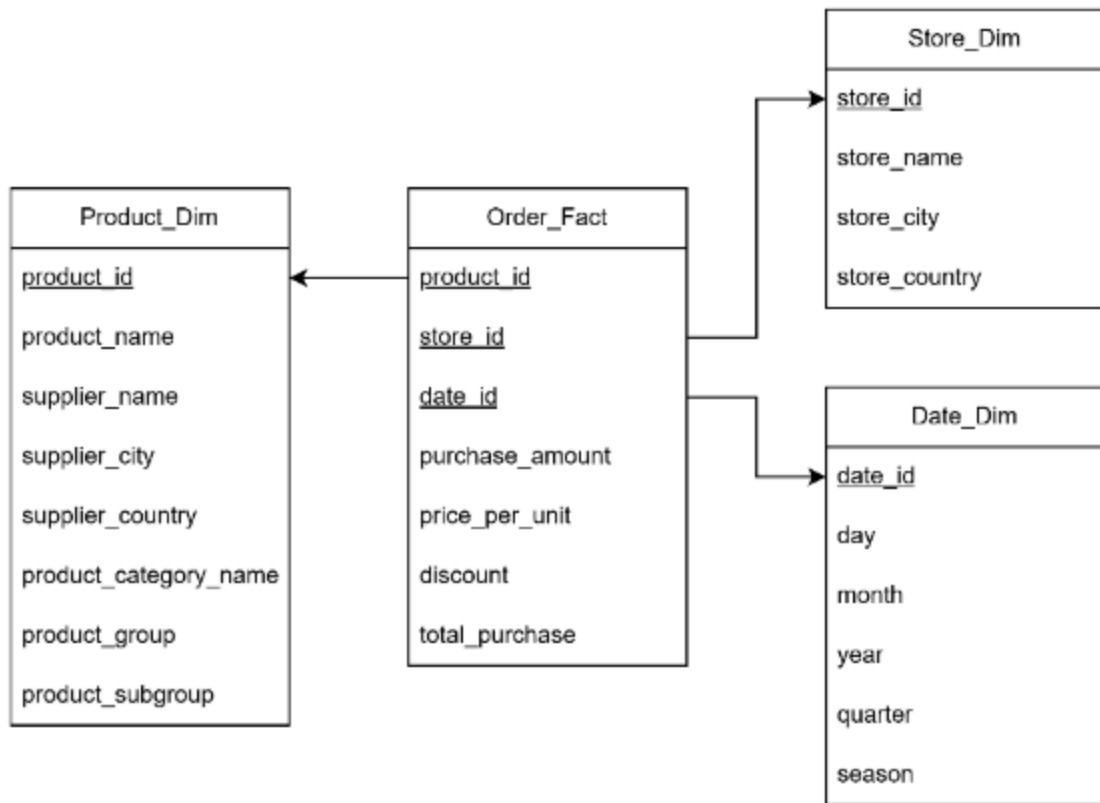


Diketahui sebuah skema basis data relasional operasional yang digunakan dalam sebuah sistem supermarket di bawah ini.

*The following is an operational relational database schema that is used for a supermarket system.*



Untuk keperluan analisis bisnis yang dilakukan pihak manajemen yang biasanya terjadi pada setiap quarter (triwulan), pihak supermarket ingin membangun sebuah basisdata analitik yang dimodelkan dengan model multidimensional sebagai berikut.



Basis data analitik akan diisi berdasarkan data dari basis data operasional dengan penjelasan sebagai berikut.

1. Tabel Product\_Dim diisi berdasarkan isi tabel Product, Product\_Category, dan Supplier
2. Tabel Store\_Dim diisi berdasarkan isi tabel Store.
3. Tabel Date\_Dim diisi berdasarkan isi atribut date pada tabel Order.
4. Tabel Order\_Fact diisi berdasarkan isi tabel Order dan Order\_Details
5. Semua atribut id pada basis data analitik merupakan artificial key, yaitu atribut baru bertipe integer yang di-generate khusus untuk keperluan basis data analitik.

**Tugas Anda:**

1. Tuliskan 1 (satu) alternatif urutan yang masuk akal untuk melakukan pengisian tabel-tabel pada tabel basis data analitik di atas dan berikan penjelasan.
2. Tuliskan SQL query untuk melakukan pengisian tabel Product\_Dim berdasarkan isi tabel Product, Product\_Category, dan Supplier. Asumsikan sudah dilakukan ekstraksi data tabel Product, Product\_Category, dan Supplier dengan struktur dan nama tabel yang sama dan tabel-tabel tersebut berlokasi di basis data yang sama dengan basis data analitik.
3. Jelaskan apakah tepat jika proses integrasi data dari basis data operasional ke basis data analitik di atas dilakukan secara batch processing.

1. Urutan yang masuk akal untuk pengisian tabel pada basis data analitik:

- a. Date\_Dim → Karena tabel ini independen dan hanya bergantung pada atribut date dari tabel Order
- b. Store\_Dim → Karena tabel ini independen dan hanya bergantung pada tabel Store
- c. Product\_Dim → Karena membutuhkan data dari tabel Product, Product\_Category, dan Supplier
- d. Order\_Fact → Harus diisi terakhir karena merupakan tabel fakta yang memerlukan foreign key dari semua tabel dimensi (date\_id, store\_id, product\_id)

Urutan ini memastikan integritas referensial terjaga, dimana tabel dimensi harus diisi terlebih dahulu sebelum tabel fakta karena tabel fakta memerlukan foreign key dari tabel-tabel dimensi.

2. SQL Query untuk mengisi tabel Product\_Dim:

```
INSERT INTO Product_Dim (  
    product_id,  
    product_name,  
    supplier_name,  
    supplier_city,  
    supplier_country,  
    product_category_name,
```

```

product_group,
product_subgroup
)

SELECT
    p.product_code,
    p.name,
    s.name,
    s.city,
    s.country,
    pc.name,
    pc.group,
    pc.subgroup
FROM
    Product p
    JOIN Supplier s ON p.supplier_code = s.supplier_code
    JOIN Product_Category pc ON p.product_category_code =
pc.product_category_code;

```

3. Penggunaan batch processing dalam proses integrasi data dari basis data operasional ke basis data analitik sangat efektif dengan alasan:
  - Data warehouse umumnya diperbarui secara berkala (dalam hal ini setiap tiga bulan) dan tidak memerlukan pembaruan secara real-time.
  - Volume data yang besar yang perlu dipindahkan seringkali membutuhkan waktu pemrosesan yang cukup lama.
  - Batch processing memfasilitasi validasi data yang lebih akurat sebelum data tersebut diintegrasikan ke dalam data warehouse.
  - Proses ETL (Ekstrak, Transformasi, Muat) bisa dilaksanakan di luar jam puncak untuk mengurangi pengaruhnya terhadap kinerja sistem operasional.
  - Proses ini juga membantu dalam mengelola kesalahan dan melakukan pemulihan data jika terjadi kesalahan selama integrasi.

3. → Proses integrasi data dari basis batch-processing karena ada pernyataan bahwa basis data yang dilakukan oleh pihak manajemen yang biasanya terjadi pada setiap quarter (triwulan). Dengan pemindahan data dari basis data operasional dapat dilakukan secara periodik, misalnya setiap 3 bulan sekali pada 1 minggu menjelang penggunaan basis data untuk keperluan analisis. Karena integrasi data hanya dilakukan per 3 bulan, terdapat high latency antara isi data basis data analitik dibandingkan dengan basis data operasional. Namun hal ini tidak menjadi persoalan karena data baru hanya dibutuhkan setiap 3 bulan.

Solution Alternative:

1. → The order of filling the tables is determined based on the foreign key references of the tables in the target database. The Order\_Fact table must be filled last because this table refers to all other dimension tables. Since there is no dependency between the dimension tables (Product\_Dim, Store\_Dim, Date\_Dim), the order of filling these tables is free.

Alternative filling sequence:

Product\_Dim\_Table → Store\_Dim\_Table → Date\_Dim\_Table → Order\_Fact\_Table

2. → Query to fill the Product\_Dim\_table,

```
INSERT INTO Product_Dim (product_name, supplier_name, supplier_city, supplier_country, product_category_name, product_group, product_subgroup)
SELECT p.name, s.name, s.city, s.country, pc.name, pc.group, pc.subgroup
FROM Product p INNER JOIN Product_Category pc USING (product_category_code)
INNER JOIN Supplier s USING (supplier_code)
-- Assumption: product_id will be generated automatically by the system
```

**Apakah schedule berikut ini legal untuk protokol konkurensi validation-based? Jelaskan jawaban Anda.**  
 Asumsikan bahwa setiap transaksi dimulai tepat sebelum instruksi pertama transaksi tersebut dieksekusi.

*Is the schedule below legal for validation-based concurrency protocols? Explain your answer. Assume that each transaction starts just before the first instruction of the transaction is executed.*

R2(B); R1(A); R2(C); R1(B); W2(C); C2; R4(B); R3(B); W1(D); W1(A); C1; R4(D); R3(C); C3; W4(B);  
 W4(D); C4;

Operation	Transac tion	Read Set	Write Set	StartTS	ValidationTS	FinishTS	Validation Check
R1(A)	T1	{A}		1	8	9	-
R2(B)	T2	{B}		2	5	6	Aman karena tidak ada write sebelum commit T2.
R2(C)	T2	{B, C}		2	5	6	Aman karena tidak ada write sebelum commit T2.
R1(B)	T1	{A, B}		1	8	9	-
W2(C)	T2	{B, C}	{C}	2	5	6	-
C2	T2	{B, C}	{C}	2	5	6	T2 berhasil commit.
R3(B)	T3	{B}		3	12	13	-
R4(B)	T4	{B}		4	15	16	-
W1(D)	T1	{A, B}	{D}	1	8	9	-

W1(A)	T1	{A, B}	{D, A}	1	8	9	-
C1	T1	{A, B}	{D, A}	1	8	9	T1 berhasil commit.
R4(D)	T4	{B, D}		4	15	16	-
R3(C)	T3	{B, C}		3	12	13	-
C3	T3	{B, C}		3	12	13	T3 berhasil commit.
W4(B)	T4	{B, D}	{B}	4	15	16	-
W4(D)	T4	{B, D}	{B, D}	4	15	16	-
C4	T4	{B, D}	{B, D}	4	15	16	T4 berhasil commit.

T1 → Berhasil commit tanpa conflict karena hanya write setelah read dan sebelum transaksi lain mengeksekusi data yang sama. Tidak ada transaksi lain yang mengubah data yang di-read atau di-write oleh T1 sebelum T1 melakukan commit.

T2 → Berhasil commit dengan write yang tidak conflict dengan read sebelumnya. T2 melakukan penulisan pada data C setelah read dan tidak ada transaksi lain yang write C sebelum T2 commit.

T3 → Tidak ada transaksi lain yang melakukan write pada B dan C yang mengganggu read T3.

T4 → Write divalidasi terhadap fase read dan write dari transaksi sebelumnya. T4 melakukan write pada B dan D setelah memastikan tidak ada transaksi lain yang read atau write data tersebut yang bisa menyebabkan konflik selama eksekusi T4.

### Kesimpulan

Transaksi dalam schedule tersebut dapat dijalankan tanpa ada rollback atau abort.

Soal 2. Validation-based Protocol¶

Apakah schedule berikut ini legal untuk protokol konkurensi validation-based? Jelaskan jawaban Anda. Asumsikan bahwa setiap transaksi dimulai tepat sebelum instruksi pertama transaksi tersebut dieksekusi.¶  
R2(B); R1(A); R2(C); R1(B); W2(C); C2; R4(B); R3(B); W1(D); W1(A); C1; R4(D); R3(C); C3; W4(B); W4(D); C4;¶

**SOLUSI**

Validasi-pertama oleh T2. Karena belum ada transaksi lain yang commit sebelum T2 maka validasi sukses dan T2 commit.¶ Validasi-kedua oleh T1. Karena T1 pernah sama-sama aktif dengan T2 yang sudah commit, maka perlu diperiksa apakah write-set T2 beririsan dengan read-set T1.  $WS(T2) = \{C\}$ ;  $RS(T1) = \{A, B\}$  → tidak beririsan sehingga validasi sukses dan T1 commit.¶ Validasi-ketiga oleh T3. Karena T3 pernah sama-sama aktif dengan T1 yang sudah commit, maka perlu diperiksa apakah write-set T1 beririsan dengan read-set T3.  $WS(T1) = \{A, D\}$ ;  $RS(T3) = \{B, C\}$  → tidak beririsan sehingga validasi sukses dan T3 commit.¶ Validasi-keempat oleh T4. Karena T4 pernah sama-sama aktif dengan T1 dan T3 yang sudah commit, maka perlu diperiksa apakah write-set T1 dan T3 beririsan dengan read-set T4.  $WS(T1, T3) = \{A, D\}$ ;  $RS(T4) = \{B, D\}$  → terdapat irisan sehingga validasi gagal.¶

Karena ada validasi yang gagal, maka schedule pada soal tidak legal jika menggunakan validation-based protocol.¶

English:¶

The first validation by T2. Since there are no other transactions committed before T2, the validation is successful and T2 commits.¶

The second validation by T1. Since T1 was once active with T2 which has committed, it is necessary to check whether T2's write-set overlaps with T1's read-set.  $WS(T2) = \{C\}$ ;  $RS(T1) = \{A, B\}$  → does not overlap so the validation is successful and T1 commits.¶

Berikut ini adalah urutan kedatangan instruksi-instruksi dari transaksi Tx, Ty, dan Tz ke DBMS.

Transaksi dimulai tepat sebelum instruksi pertama pada transaksi tersebut. Jika dibutuhkan, timestamp  $(Tx,Ty,Tz) = (1,2,3)$ .

*The following is the order of arrival of instructions from transactions Tx, Ty, and Tz to the DBMS.*

*The transaction starts just before the first instruction in the transaction. If needed, timestamp  $(Tx,Ty,Tz) = (1,2,3)$ .*

Tx	Ty	Tz
R(A)		
W(A)		
	R(B)	
		R(B)
		W(B)
R(B)		
W(B)		
commit		
	R(C)	
commit		
	R(C)	
	W(C)	
	commit	

**Tuliskan** langkah-langkah untuk mengeksekusi urutan instruksi tersebut dengan timestamp ordering protocol sampai semua transaksi tuntas dieksekusi dan **jelaskan** setiap tahapan eksekusi. Jika terjadi abort untuk suatu transaksi, DBMS akan memprioritaskan mengeksekusi ulang dari instruksi transaksi tersebut sampai bagian instruksi yang menyebabkan abort.

Di awal:

$$TS(A) = (0,0); TS(B) = (0,0); TS(C) = (0,0)$$

Tx (TS = 1)	Ty (TS = 2)	Tz (TS = 3)	
R(A)			Executed karena $1 > 0$ ; $TS(A) = (1,0)$
W(A)			Executed karena $1 > 0$ dan $1 > 0$ $TS(A) = (1,1)$
	R(B)		Executed karena $2 > 0$ ; $TS(B) = (2,0)$
		R(B)	Executed karena $3 > 0$ $TS(B) = (3,0) \rightarrow$ ambil 3 karena $\max(2,3)$
		W(B)	Executed karena $3 = 3$ dan $3 > 0$ $TS(B) = (3,3)$
R(B)			Tx rolled-back karena $1 < 3$
Ax			Tx dijalankan kembali dengan timestamp

			baru (misal 10)
R(A)			Executed karena $10 > 1$ TS(A) = (10,1) → ambil 10 karena max(10,1)
W(A)			Executed karena $10 = 1$ dan $10 > 1$ TS(A) = (10,10)
R(B)			Executed karena $10 > 3$ ; TS(B) = (3,10)
W(B)			Executed karena $10 > 3$ dan $10 = 10$ TS(B) = (10,10)
commit			
	R(C)		Executed karena $2 > 0$ TS(C) = (2,0)
	commit		
		R(C)	Executed karena $3 > 0$ TS(C) = (3,0) → ambil 3 karena max(2,3)
		W(C)	Executed karena $3 = 3$ dan $3 > 0$ TS(C) = (3,3)
		commit	

Office Update To keep up-to-date with security updates, fixes and improvements, choose Check for Updates.

i. → Multiverion timestamp ordering protocol.  
ii. → Multiverion two-phase locking protocol.  
iii. → Snapshot isolation menggunakan aturan first committer wins.

**Soal-a-Timestamp-Ordering-Protocol**

$(x,y) = (RTS, WTS)$

Tx=1	Ty=2	Tz=3	A	B	C	Note
R(A)			(1,0)			
W(A)			(1,1)			
	R(B)			(2,0)		
		R(B)		(3,0)		
		W(B)		(3,3)		
R(B)				(3,3)		TS(Tx) < WTS(B), rejected
Abort						TS(Tx) = 4
Tx=4						
R(A)			(4,0)			
W(A)			(4,4)			
R(B)				(4,3)		
W(B)				(4,4)		
commit						
	R(C)					(2,0)
	commit					(3,0)
		R(C)				(3,3)
		W(C)				
		commit				

**Soal-b.i-Multiverion-Timestamp-Ordering-Protocol**

$<Q_x, RTS, WTS> = \text{data } Q \text{ version } x \text{ with R-timestamp } = \text{RTS} \text{ and W-timestamp } = \text{WTS}$

Initial Data Version: 0  
General All Employees (unrestricted)

Berikut ini adalah urutan kedatangan instruksi-instruksi dari transaksi Tx, Ty, dan Tz ke DBMS. Transaksi dimulai tepat sebelum instruksi pertama pada transaksi tersebut. Jika dibutuhkan, timestamp  $(Tx, Ty, Tz) = (1, 2, 3)$ .

*The following is the order of arrival of instructions from transactions Tx, Ty, and Tz to the DBMS. The transaction starts just before the first instruction in the transaction. If needed, timestamp  $(Tx, Ty, Tz) = (1, 2, 3)$ .*

Tx	Ty	Tz
R(A)		
W(A)		
	R(B)	
	R(B)	
	W(B)	
R(B)		
W(B)		
commit		
	R(C)	
	commit	
	R(C)	
	W(C)	
	commit	

Pilihlah salah satu dari protokol berikut (**tuliskan dengan jelas pilihan dalam jawaban**) kemudian sebutkan dan jelaskan apakah urutan kedatangan instruksi tersebut dapat dijalankan dengan protokol tersebut. Berikan penjelasan untuk setiap tahapan eksekusi sampai seluruh transaksi sukses berjalan atau sampai salah satu abort.

*Choose one of the following protocols (clearly state the choice in the answer) then state and explain whether the instruction arrival sequence can be executed with that protocol. Provide an explanation for each stage of execution until all transactions successfully run or until one aborts.*

1. Multiversion timestamp ordering protocol.
2. Multiversion two phase locking protocol.
3. Snapshot isolation menggunakan aturan *first committer wins*.

### Multiversion timestamp ordering

Di awal:

$\langle A_0, 0, 0 \rangle; \langle B_0, 0, 0 \rangle; \langle C_0, 0, 0 \rangle; \langle D_0, 0, 0 \rangle$

<b>Tx (TS = 1)</b>	<b>Ty (TS = 2)</b>	<b>Tz (TS = 3)</b>	
R(A) <A0,1,0>			T1 gets <A0, 0, 0> and updates A0
W(A) <A1,1,1>			T1 gets <A0, 1, 1> and create A1
	R(B) <B0,2,0>		T2 gets <B0, 0, 0> and updates B0
		R(B) <B0,3,0>	T3 gets <B0,2,0> and updates B0
		W(B) <B1,3,3>	T3 gets <B0,3,0> and create B1
R(B) <B0,3,0>			T1 gets <B0,3,0> and updates B0
W(B)			Tx rolled-back karena TSx < W-TS(B1) (1<3)
Ax			

Schedule tidak dilanjutkan karena Tx sudah aborted