BAB II – Index

- Bitmap Index Datawarehouse
    - Bitmap index, digunakan pada kolom yang memiliki varian nilai yang sedikit, semisal jensi kelamin (Laki/Perempuan), status nikah (Lajang/Menikah), dll
    - Selain itu, cocok digunakan untuk kolom yang mengandung nilai NULL
    - Namun, bitmap index hanya cocok digunakan pada local partitioning index
    - Dapat digunakan pada tabel join yang dinamakan Bitmap Join Indexes (star skema datawarehouse penjualan, tabel sales join dengan tabel customer)
    - Keuntungan
        - Mengurangi waktu respon untuk query data yang besar
        - Mengurangi alokasi penyimpanan untuk index dibandingkan index yang lain
        - Bekerja secara minimum di hardware (penggunaan CPU rendah, dan memori yang kecil)
        - Maintenance dapat dilakukan secara efisien selama dilakukan proses DML dan loading data

- B-Tree Index Datawarehosue
    - Index berbentuk tree, dan analoginya seperti pada saat mencari topik pada sebuah buku melalui daftar isi.
    - Cocok digunakan pada sebuah kolom yang memiliki rentang nilai yang besar/banyak (nama, tanggal lahir, alamat, dll)
    - B-Tree Index dapat bekerja pada local maupun global partitioning index

- Index Compression
    - Index compression akan secara otomatis dilakukan oleh oracle pada saat pembuatan index Bitmap atau B-Tree
    - Tujuan dari index compression adalah untuk efisiensi storage, namun akan meningkatkan kinerja CPU

- Local Index vs Global Index
    - Pada oracle versi 8i atau sebelumnya, penggunaan global index tidak direkomendasikan dalam data warehouse
        - Hal ini dikarenakan, pada statement DDL dalam partisi (ALTER TABLE, DROP PARTITION, dll), dapat mengakibatkan semua index tidak valid lagi, sehingga harus membangun index dari awal, yang tentunya langkah ini tidak efisien
    - Pada oracle versi 10g, ada perubahan pada global index, sehingga global index dapat dimaintenance setelah menggunakan DDL. Sehingga validasi index lebih mudah dilakukan dan membuat global index lebih efektif untuk data warehouse
    - **NAMUN,** meskipun terdapat perubahan, local index lebih umum digunakan daripada global index.
- **TIPS PEMBUATAN INDEX DATA**
    - Buat index hanya pada baris record yang besar (>1000 record)
    - Buat index hanya pada kolom yang digunakan pada klausa SQL where
    - Contoh pada tabel **emp** pada hr schema

        - SELECT * FROM EMP WHERE ENAME='Robert';
        - Yang harus di index adalah pada kolom ENAME
        - Karena ename adalah data dengan range yang lebar (nama) maka index yang digunakan adalah B-Tree Index
    - Contoh yang lain
        - SELECT * FROM EMP EHERE ENAME='Robert' AND deptno=40;
        - Yang harus dibuat indexnya adalah **index gabungan** antara ename dan deptno
    - Contoh yang lain
        - SELECT * FROM EMP WHERE ENAME='Robert' or deptno=40;
        - **Index harus dibuat terpisah** yaitu index kolom ename dan index kedua adalah index kolom deptno

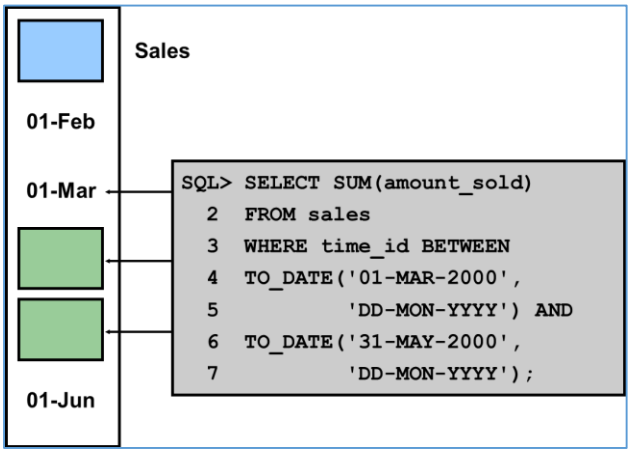
BAB III – TUNING DATA WAREHOUSE

- Performa Input dan output dalam data warehouse harus selalu menjadi pertimbangan utama pada saat melakukan desain data warehouse.
- Beban kerja yang harus dihadapi oleh data warehouse antara lain
    - Input output yang intensif karena loading data berjumlah besar
    - Pada saat membuat index
    - Pada saat membuat materialized view
    - Dan pada saat melakukan query dengan volume data yang besar
- Pada tuning data warehouse, ada beberapa hal yang perlu diperhatikan, antara lain
    - Mengkonfigurasi I/O bandiwth
        - Mengkonfigurasikan storage dalam I/O bandwith lebih penting dibandingkan dengan menambah kapasitas storage nya
        - Yang dikonfigurasikan pada I/O bandwith antara lain
            - I/O throughput rates
            - RPM
            - Buffer cache
            - Write dan Read Speed
    - Membagi Storage
        - Dalam mengkonfigurasikan storage, secara umum untuk setiap object database yang ada disimpan dalam beberapa disk dengan akses channel yang berbeda beda
        - Dalam oracle, pembagian storage ini dapat dilakukan dengan membuat beberapa datafiles dalam database oracle
        - Datafiles tersebut nantinya dapat didistribusikan ke beberapa disk yang berbeda-beda
    - Menggunakan Storage Redundancy sebagai backup
        - Karena jumlah data pada data warehouse sangat masif, maka storage yang dibutuhkan akan sangat besar, sehingga sangat memungkinkan terjadinya kerusakan storage. Untuk mengatasi hal tersebut sebaiknya dibuat juga disk redundancy sebagai alat backup data pada datawarehouse

        - Setiap disk master (storage master) setidaknya memiliki satu disk backup (disk redudancy)
        - Teknologi yang sering digunakan adalah RAID untuk membuat disk redudancy
    - Test sistem I/O sebelum membuat database
        - Sekali file database dibuat, akan sangat sulit mengkonfigurasikannya kembali, maka harus dilakukan testing sistem I/O terlebih dahulu sebelum database dibuat.
        - Hal ini dilakukan untuk menghindari adanya pembangunan ulang database yang disebabkan adanya keharusan untuk me-rekonfigurasi ulang I/O sistemnya
    - Merencanakan desain untuk pertumbuhan data kedepannya
        - Dalam pembuatan data warehouse, wajib untuk designer datawarehouse untuk memiliki pandangan terhadap pertumbuhan data kedepannya, hal ini dilakukan untuk meningkatkan efisiensi dan durability dari suatu data warehouse

BAB III – PARTISI

- Partition merupakan fungsionalitas yang memungkinkan table, index dan index-organized table dibagi menjadi bagian yang lebih kecil, sehingga dapat menjadikan object database dapat di manage dan diakses secara granular.
- Keuntungan yang didapat dengan menggunakan partitioning
    - Partition dapat di scan, update, insert atau delete secara paralel
    - Operasi join dapat dioptimasi dengan join 'by the partition'
    - Partition dapat disebar agar bebannya seimbang (load-balanced) ke sepanjang physical devices yang ada
    - Optimizer menghilangkan partition yang tidak butuh untuk di scan (pruning/pemangkasan)
- Partition prunning merupakan metode untuk mengakses hanya beberapa data partisi sesuai kondisi yang diberikan



- Tabel vs. Index Partitioning
    - Tabel yang tidak dipartisi (nonpartitioned table) bisa memiliki partisi atau nonpartition index
    - Tabel yang sudah dipartisi juga dapat memiliki partisi atau nonpartition index

- Partitioning methods
    - **Range partitioning**, memetakan data menjadi partisi atas dasar range dari nilai partition key untuk setiap partisi. Range partition menggunakan range nilai dari kolom untuk memetakan row data ke partisi partisi. Rangenya harusnya memiliki batas atas dan batas bawah. Contohnya adalah partisi file sales berdasarkan tanggal atau berdasarkan waktu.
    - **Hash partitioning,** memetakan data menjadi partisi dengan menggunakan *hashing alogirtm* yang diaplikasikan ke *partition key.* Hashing algorithm membagi rata baris ke masing-masing partisi, sehingga partisi memiliki ukuran hampir sama.
    - **List partitioning,** memetakan baris menjadi partisi dengan menggunakan daftar *discrete values* yang digunakan untuk mempartisi kolom. Dengan metode ini, kontrol bisa didapatkan perihal bagaimana baris data di petakan ke partition. Contohnya (mungkin seperti ini) melakukan partisi data penjualan, berdasarkan kolom kota, atau kolom provinsi, dan lain lain
    - **Composite partitioning,** menggabungkan range dengan hash atau list partitioning. Data akan didistribusikan ke partisi sesuai dengan batas yang dibuat oleh *partition range*.
        - Range-hash, melakukan subpartition pada range partition menggunakan *hashing algorithm*
        - Range-list, melakukan subpartition pada range partition menggunakan *explicit list*
- Pembuatan table menggunakan partition. Deklarasinya memiliki tiga elemen
    - Struktur logikal dari table (logical structure)
    - Struktur partisi yang ingin digunakan, dengan cara mendefinisikan tipe dan kolomnya
    - Struktur tiap partition table
        - Logical bound
        - Physical storage attributes

```
SQL> CREATE TABLE example
  2   ( idx NUMBER, txt VARCHAR2(20) )
  3   PARTITION BY RANGE ( idx )
  4   (  PARTITION VALUES LESS THAN ( 0 )
  5        TABLESPACE data01
  6   ,  PARTITION VALUES LESS THAN ( MAXVALUE ));
```

- Local attribute :
    - Normal table structure (columns, constraint)
    - Partition type
    - Key & Value
    - Row movement
- Physical attribute :
    - Tablespace
    - Extent size, block attribute
- Equipartitioning
    - Jika dua tabel memiliki partition key dan partition key value yang sama, maka bisa dikatakan tabel tersebut merupakan equipartitioned
    - Sangat berguna untuk tabel yang memiliki common key (key yang pasaran), seperti relasi master-detail (misal tabel-user & tabel-detail-user)

    - Operasi partitionwise join membutuhkan adanya equipartitioning
    - Index dapat di equipartitioned dengan tabel
- Full partitionwise joins
    - Full partitionwise join membagi large join menjadi small join
    - Tabel yang di join-kan harus equipartition pada key join-nya
    - Ketika full partitionwise join di eksekusi secara paralel, hasilnya akan berupa partisi
- Partial partitionwise joins
    - Partial partitionwise join hanya membutuhkan satu tabel yang terpartisi pada join key nya
    - Tabel yang terpartisi dinamai *reference* table
    - Tabel lainnya akan secara dinamis dipartisi ulang sesuai dengan partisi yang ada di reference table
    - Partial partitionwise join hanya dapat dilakukan secara paralel

BAB IV – ETL (Extraction, Transformation, Loading)

- Datawarehouse perlu untuk di update secara berkala agar dapat digunakan untuk analisis. Untuk melakukan update tersebut, harusnya data terlebih dahulu di ekstraksi dari berbagai sumber, proses ini secara umum disebut sebagai ETL. Keseluruhan proses ETL sendiri melibatkan **ekstraksi data**, **loading data**, **transportasi data**, **transformasi data** dan proses proses lainnya. Tidak jarang data yang diekstraksi memiliki perbedaan format, sehingga perlu dilakukan transformasi terlebih dahulu agar proses ETL berjalan baik. Memang **ETL merupakan singkatan dari Extraction, Transformation, Loading,** namun kenyataannya **prosesnya lebih luas dari tiga istilah tersebut**

Hal-hal yang terjadi pada saat proses ETL

- Data akan diidentifikasi dan diproses dari berbagai sumber data
- Beberapa proses transformation mungkin terjadi selama proses extraction process
- Setelah extraction, data harus di transportasikan ke sistem atarget atau sebuah sistem (intermediate system) untuk dilakukan prosesing data
- Bergantung pada metode transportation-nya beberapa transformation dapat dilakukan secara bersamaan

EXTRACTION

- Extraction memiliki dua metode
    - Logical, terdapat dua jenis
        - Full extraction
            - Semua data di pull (diambil)
            - Hanya sedikit informasi yang di track (tidak perlu menspesifikan yang mana yang akan di pull, langsung di gradak)
            - Membutuhkan waktu yang lebih lama untuk pulling data
        - Incremental extraction
            - Data di pull dalam beberapa bagian
            - Harus melakukan track data apa yang ingin di pull (harus menspesifikan data mana yang akan di pull)

        - Waktu yang dibutuhkan lebih cepat untuk pulling data
        - Dapat dlakukan dengan menggunakan CDC, CDC dapat capture dan publish data yang telah di commit dengan mode,
            - Synchronous
                - Trigger pada source database memperbolehkan data yang berubah di capture pada saat itu juga
            - Asynchrounous
                - Data yang berubah dapat di capture setelah terjadi operasi DML di commit
    - Physical, memiliki dua jenis
        - Online extraction
            - Pull data langsung dari source system
        - Offline extraction
            - Pull data yang ada di staging area
            - Staging area meliputi flat files, dump files, dan transportable tablepspace
                - Flat files, membutuhkan data yang predefined dengan generic format
                - Dump files, harus dalam format oracle
                - Redo dan archive log, data terletak pada dump files yang spesifik
                - Transportable tablespace, merupakan metode yang bagus, dan cepat untuk memindahkan data dengan volume besar
- Pilihan tentang bagaimana data dieksrak secara logical mempengaruhi cara data di ekstrak secara physical
- Untuk implementasi ekstraksi file, dapat dilihat
    - Ekstraksi data ke file
        - Menggunakan spooling dari SQLPlus
        - Menggunakan OCI atau Pro*C untuk dump file
        - Menggunakan Data Pump untuk meng-export data ke oracle dump file
        - Menggunakan external tables
    - Ekstraksi melalui distributed operation, dengan menggunakan teknologi ini, database oracle dapat secara langsung melakukan query table yang lokasinya ada di bermacam-macam source system
- Hal-hal yang perlu diperhatikan pada saat menentukan method adalah sebagai berikut
    - Business needs (Mempertimbangkan kebutuhannya)
    - Lokasi dari source dan target system
    - Ketersediaan (availabbility) dari source system
    - Waktu yang diperlukan untuk mengekstrak data

TRANSPORTATION

- Terdapat tiga pilihan dasar yang dapat dipilih pada saat melakukan transportation
  - Transportation menggunakan flat files
    - Dengan metode ini, data disimpan dalam bentuk file dan dikirimkan melalui metode pengiriman standar, ftp, secara fisik, download atau yang lainnya. Memungkinkan untuk mentransfer data antar system yang berbeda
    - Tingkat kegagalan baru akan diketahui pada saat flat file di proses untuk dimasukan ke target system
    - Sangat efisien dan efektif jika transportasinya antara system yang sama
  - Transportation melalui distributed operation
    - Metode transportation ini memungkin target system secara langsugn mengakses data yang ada pada source system dengan cara query secara langsung.
    - Tingkat kegagalan dapat diketahui secara langsung dari hasil query yang dilakukan
  - Transportation menggunakan transportable tablespaces
    - Dengan menggunakan transportable tablespace, data files (yang berisi table, index dan semua object database yang ada) dapat secara langsung di transfer ke database lainnya. Dapat dikatakan hanya sekadar export & import database
    - Merupakan metode tercepat untuk perihal memindahkan data yang sangat besar
    - Source dan target database bisa memiliki block size yang berbeda
    - Sangat berguna untuk mentransfer data dari OLTP ke data warehouse,
    - Namun, sebelum oracle 10g, database source dan target harus menggunakan OS yang sama

LOADING

- Terdapat beberapa mekanisme data-loading, yaitu
  - SQL *Loader
    - Akan melakukan load data yang ada di flat file ke tabel yang sudah ada
    - Dapat melakukan operasi transformasi sederhana ketika loading
    - Irect-path loading bisa digunakan untuk mengurangi load time
    - Ketika menggunakan metode ini, data pada flat file tidak dapat diakses (not accessible) sampai dengan data selesai di load
  - External Tables
    - External tables merupakan read-only tables dimana data disimpan di luar database tepatnya di flat file
    - Data dapat di query (seperti virtual table) menggunakan supported language pada database

- Tidak boleh ada operasi DML dan tidak satupun index dapat dibuat
- Mendeskripsikan bagaimana external data harus ditampilkan/direpresentasikan pada database
- Sangat berguna pada environment dimana external source harus di-join dengan obje database lain lalu harus ditransformasi
- Dan sangat berguna ketika external data berukuran sangat besar dan tidak sering di query

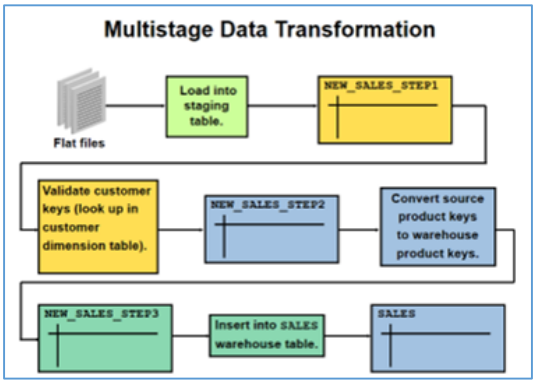
▪ OCI dan direct-path APIs
- Memungkinkan untuk transformation dan loading pada waktu yang bersamaan
- Source system diakses secara online
- Tidak perlu tahapan perantara seperti flat file

▪ Export/Import
- Sangat efektif untuk small loads
- Memudahkan transfer antara database oracle dengan OS yang berbeda-beda

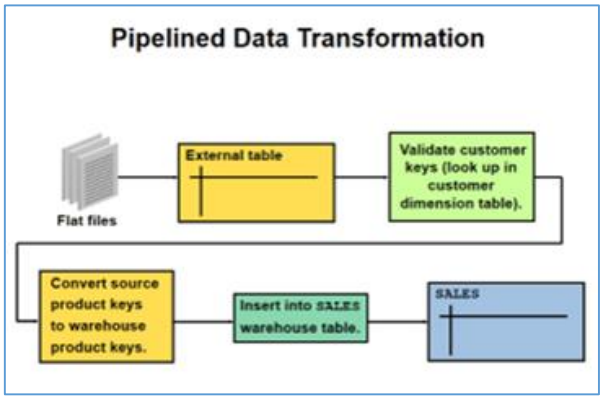
▪ Data Pump

TRANSFORMATION

- Data transformation merupakan tahapan paling kompleks dan paling *costly* (mahal) dari proses ETL
- Transformation dapat berupa hanya konversi data sederhana sampai dengan operasi yang lebih kompleks
- Banyak operasi transformation dapat dilakukan di database oracle
- Data dapat di transformasi dengan dua cara
  - Multistage
    - Flat file di load ke staging table
    - Terbentuk table (contoh : NEW_SALES_S1)
    - Validasi key nya dengan cara melihat dimension tablenya
    - Terbentuk table kedua (contoh : NEW_SALES_S2)
    - Konversi source key sesuaikan dengan key yang ada pada warehouse
    - Terbentuk table ketiga (contoh : NEW_SALES_S3)
    - Insert data yang sudah di konversi ke tabel yang ada di warehouse
    - Jadilah data baru di tabel (contoh : SALES)



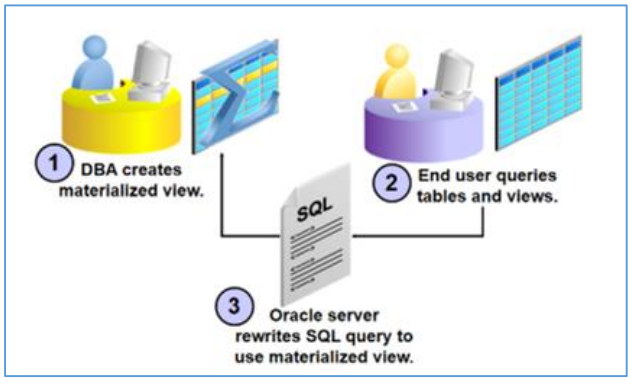
- Pipeline
  - Flat files, berisi external table
  - Validasi key nya dengan melihat dimension tablenya
  - Lalu konversi source key sesuai dengan key yang ada pada warehosue
  - Insert pada tabel di warehouse
  - Data baru di insert pada tabel warehouse



- Transformasi dapat dilakukan dengan cara cara berikut ini
  - SQL
    - Setelah data di load ke database, transformation dapat dieksekusi dengan operasi,
      - CREATE TABLE ... AS SELECT
      - INSERT /*+APPEND*/ AS SELECT
      - UPDATE
      - MERGE
        - Statemen ini memungkinkan untuk melakukan kondisional UPDATE/INSERT
        - Kondisinya di spesifikan pada klausa ON
        - Statement ini akan melakukan UPDATE jika row sudah ada, dan INSERT jika row belum ada
      - Multitable INSERT
        - Dapat digunakan pada system data warehouse untuk mentransfer data dari satu atau lebih source
  - PL/SQL
  - Table function
    - Table function merupakan function yang dapat menghasilkan beberapa baris sebagai output (inputnya juga dapat berupa baris)
    - Table function mendukung pipelined dan parallel execution menggunakan PL/SQL, C, atau Java
- DML Error Logging
  - Membatalkan bulk DML operation yang sedang berjalan membutuhkan waktu yang lama dan resource yang besar
  - Dengan DML error logging operasi bulk DML dapat berlanjut prosesnya meskipun error terjadi
  - Error akan disimpan pada error-logging table

BAB V – Materialized View

- Materialized view adalah object database yang berisi hasil dari sebuah query
- Untuk menjelaskan urgensitas materialized view, berikut adalah statement sebelumnya
  - Bagaimana caranya meningkatkan query response time ?
    - Menggunakan index
    - Partisi datanya
    - Implementasikan paralel execution
  - NAMUN, bagaimana dengan proses precomputing query-nya ?
    - Dengan cara membuat ***summary*** yaitu **Materialized Views**
    - Akan secara otomatis rewrite SQL app nya
    - Dan akan secara otomatis refresh datanya
- Menggunakan **summary** untuk meningkatkan prforma
  - Dapat meningkatkan query execution dengan cara precalculating expensive join dan aggregation operations sebelum diekseskusi DAN hasilnya disimpan di sebuah tabel
  - Cara membuatnya adalah dengan membuat materialized view
- Materialized view dapat digunakan sebagai summary management. Pada prakteknya, tahapannya adalah sebagai berikut
  - DBA membuat materialized view
  - Ketika user melakukan query table dan view, mekanisme *query rewrite* dari oracle akan secara otomatis diieksekusi dan akan me-rewrite SQL query yang digunakan untuk membuat summary table



- Summary management memiliki beberapa komponen, yaitu
  - Mekanisme untuk mendefinisikan materialized view
  - Refresh mechanism untuk menjamin bahwa materialized view berisi data yang update
  - Query rewrite untuk melakukan reqrite query ke materialized view
- Materialized view dapat dibuat lebih dari satu, namun berapa yang harus dibuat agar bekerja secara efektif ?
  - Satu materialized view per query
    - Sangat ideal untuk meningkatkan performa query
    - Namun memerlukan disk space yang besar
    - Tidak disarankan
  - Satu materialized view untuk beberapa query

        - Satu materialized view dapat digunakan untuk memenuhi banyak query
        - Disk space tidak terlalu besar
        - Dan lebih sedikit waktu yang dibutuhkan untuk memaintain materialized view
    - Jadi yang mana yang harus dipilih ?
        - Sesuai dengan workload yang dimiliki
        - Dapat dipertimbangkan menggunakan SQL access advisor
    - Refresh option materialized view
        - Materialized view dapat di-refresh dengan mode
            - ON DEMAND : manual
            - ON COMMIT : Refresh terjadi ketika transaction commit
        - Materialized view dapat direfresh dengan cara
            - COMPLETE
            - FAST
            - FORCE
            - NEVER

PERTANYAAN

- Nomor 1
    - Pengertian ETL, klik
    - Ekstraksi Logika dan Fisik, klik
    - Tipe Transformasi, klik
- Nomor 2
    - Pengertian dan manfaat partisi, klik
    - Jenis partisi tabel atau partisi index, klik
    - Rekomendasi partisi dengan SQL berdasarkan kota

```
CREATE TABLE sales_list
(salesman_id NUMBER(5),
salesman_name VARCHAR2(30),
sales_state VARCHAR2(20),
sales_amount NUMBER(10),
sales_date DATE)
PARTITION BY LIST(sales_state)
(PARTITION sales_west VALUES('California',
'Hawaii'),
PARTITION sales_east VALUES ('New York',
'Virginia', 'Florida'),
PARTITION sales_central VALUES('Texas',
'Illinois'),
PARTITION sales_other VALUES(DEFAULT));
```

- Nomor 3
    - Pengertian dan manfaat materialized view, klik
    - Kegunaan query rewrite
        - Fitur untuk secara otomatis menulis ulang(rewrite) SQL query yang dieksekusi oleh user dengan SQL query yang lain. Dalam penerapan materialized view, query rewrite akan me-rewrite query user menjadi query SQL untuk menggunakan summary tables

- Contohnya adalah, jika user melakukan query

```
SELECT c.cust_id, SUM(amount_sold)
FROM   sales s, customers c
WHERE  s.cust_id = c.cust_id
GROUP BY c.cust_id;
```

- Lalu di rewrite oleh oracle menggunakan query rewrite menjadi

```
SELECT * FROM cust_sales_mv;
```

- Syntax untuk membuat material view dengan rewrite enable,

```
CREATE MATERIALIZED VIEW cust_sales_mv
ENABLE QUERY REWRITE AS
SELECT c.cust_id, SUM(amount_sold)
FROM   sales s, customers c
WHERE  s.cust_id = c.cust_id
GROUP BY c.cust_id;
```