Materializations

- In some cases it is possible to achieve a better performance via enforcing materializations of specific subselects

- An abstract example:

```
SELECT * FROM T1 JOIN T2 JOIN T3 JOIN T4 JOIN ...
```

- Assumptions:
  - (`T2 JOIN T3 JOIN T4`) has a small result set (e.g. due to strong local filter)
  - Optimizer is unable to evaluate (`T2 JOIN T3 JOIN T4`) first (e.g. `OUTER JOIN`)
- Pre calculation of (`T2 JOIN T3 JOIN T4`) by enforcing the materialization of it

Example: materialization

- Example (Modified query 2 from Profiling exercise):

```
SELECT ma.MARKET_ID, ma.CITY, max(amount) MAX_ARTICLES_SOLD,
       avg(amount) AVG_ARTICLES_SOLD
FROM MARKETS ma
-- left join to retrieve markets without SALES
LEFT JOIN SALES s
  ON ma.MARKET_ID = s.MARKET_ID
JOIN SALES_POSITIONS sp
  ON s.SALES_ID = sp.SALES_ID
WHERE s.SALES_DATE = date '2014-03-17' OR s.SALES_DATE IS NULL
GROUP BY ma.MARKET_ID, ma.CITY;
```

- Join order is predetermined due to an OUTER_JOIN:
  - SCAN MARKETS → LEFT JOIN SALES → JOIN SALES_POSITIONS
- SALES_DATE is a strong local filter ~20% → small result set
- Local filter is evaluated after the last join

Example: materialization

```
SELECT ma.MARKET_ID, ma.CITY, max(amount) MAX_ARTICLES_SOLD,
       avg(amount) AVG_ARTICLES_SOLD
FROM MARKETS ma
-- left join to retrieve markets without SALES
LEFT JOIN SALES s
  ON ma.MARKET_ID = s.MARKET_ID
JOIN SALES_POSITIONS sp
  ON s.SALES_ID = sp.SALES_ID
WHERE s.SALES_DATE = date '2014-03-17' OR s.SALES_DATE IS NULL
GROUP BY ma.MARKET_ID, ma.CITY;
```

PART_NAME	PART_INFO	OBJECT_NAME	OBJECT_ROWS	OUT_ROWS	DURATION
COMPILE / EXECUTE					2.774
SCAN	on REPLICATED table	MARKETS	4,008	1,002	0.001
OUTER JOIN	GLOBAL	SALES	322,186,828	1,008,777	11.613
JOIN	GLOBAL	SALES_POSITIONS	3,381,550,778	10,596,570	0.285
GROUP BY	GLOBAL on TEMPORARY table	tmp_subselect0	0	1,002	0.312

Materializations

- Enforcing a materialization of the local SALES_DATE filter would lead to smaller intermediate results
- Fewer rows in the last join
  - ~322 million vs. ~1 million row
- Some statements require a materialized subselect
  - `ORDER BY`
  - `ROWNUM`
  - `GROUP BY` (outer more filters may be included)
  - `DISTINCT` (might be expensive)

Example: materialization

- Materialization of local filter on SALES:

```
WITH mat_sales AS (
  SELECT MARKET_ID, SALES_ID FROM RETAIL.SALES
  WHERE SALES_DATE = date '2014-03-17'
  ORDER BY FALSE
)
SELECT ma.MARKET_ID, ma.CITY, max(amount) MAX_ARTICLES_SOLD,
       avg(amount) AVG_ARTICLES_SOLD
FROM MARKETS ma
-- left join to retrieve markets without SALES
LEFT JOIN mat_sales s
  ON ma.MARKET_ID = s.MARKET_ID
JOIN SALES_POSITIONS sp
  ON s.SALES_ID = sp.SALES_ID
GROUP BY ma.MARKET_ID, ma.CITY;
```

- `ORDER BY` enforces the materialization of the subselect
  - Constant values in `ORDER BY` clause will avoid the sorting but enforces the materialization
  - Best practice: `ORDER BY FALSE`

Example: materialization

P_NAME	P_INFO	O_NAME	OBJ_ROWS	OUT_ROWS	DURATION
COMPILE / EXECUTE					0.073
SCAN		SALES	322,186,828	1,008,777	0.042
INSERT	on TEMPORARY table	tmp_subselect1	0	1,008,777	0.021
INDEX CREATE	on TEMPORARY table	tmp_subselect1	1,008,777	1,008,777	0.036
SCAN	on REPLICATED table	MARKETS	4,008	1,002	0.000
OUTER JOIN	GLOBAL on TEMPORARY table	tmp_subselect1	1,008,777	1,008,777	0.003
JOIN	GLOBAL	SALES_POSITIONS	3,381,550,778	10,596,570	0.052
GROUP BY	GLOBAL on TEMPORARY table	tmp_subselect2	0	1,002	0.128

ORDER BY requires a materialized table.
After materialization the same join queue is used. But the materialized filter of SALES is used instead of SALES. This avoids a lot of network traffic.

Example: materialization

- We replaced an expensive global join by a much cheaper local join
- Additional costs:
  - materialization
  - index creation on a temporary table
- Run time improvement dominates the additional costs
- Attention:

  Example was executed in single user mode and the materialization was very small (400 MiB TEMP)

In this case the materialization is very small. Therefor the created overhead is small, too. If a materialization is much bigger, more DBRAM is used to keep the temporary materialization.
It will be necessary to use more resources like NET (replication) and CPU (non persitent index creation), too.