IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

DEVINE LICENSING LLC, Plaintiff, v. **SAP AMERICA, INC.,** Defendant.	C.A. No. TRIAL BY JURY DEMANDED

COMPLAINT FOR INFRINGEMENT OF PATENT

COMES NOW, Devine Licensing LLC ("Devine" or "Plaintiff"), through the undersigned attorneys, and respectfully alleges, states, and prays as follows:

NATURE OF THE ACTION

1. This is an action for patent infringement under the Patent Laws of the United States, Title 35 United States Code ("U.S.C.") to prevent and enjoin defendant SAP America, Inc. (hereinafter "Defendant"), from infringing and profiting, in an illegal and unauthorized manner and without authorization and/or of the consent from Devine, from U.S. Patent No. 6,339,769 (the "'769 patent", attached hereto as Exhibit "A") pursuant to 35 U.S.C. § 271, and to recover damages, attorney's fees, and costs.

THE PARTIES

2. Plaintiff is a Texas entity with its principal place of business at 2108 Dallas Pkwy., Suite 214-1018, Plano, Texas 75093-4362.

3. Upon information and belief, Defendant is a corporation organized under the laws of Delaware, having a principal place of business at 3999 West Chester Pike, Newtown Square, Pennsylvania 19073. Upon information and belief, Defendant may be served with process at

The Corporation Trust Company, Corporation Trust Center, 1209 Orange Street, Wilmington, Delaware 19801.

JURISDICTION AND VENUE

4. The Court has subject matter jurisdiction over this action pursuant to 28 U.S.C. §§ 1331 and 1338(a) because the action arises under the Patent Laws of the United States, 35 U.S.C. §§ 1 et seq.

5. This Court has personal jurisdiction over Defendant by virtue of its systematic and continuous contacts with this jurisdiction, including residing in Delaware, as well as because of the injury to Devine, and the cause of action Devine has risen, as alleged herein.

6. Defendant is subject to this Court's specific and general personal jurisdiction pursuant to due process and/or the Delaware Long Arm Statute, *Del. Code. Ann. Tit. 3, § 3104*, due at least to its substantial business in this forum, including: (i) at least a portion of the infringements alleged herein; and (ii) regularly doing or soliciting business, engaging in other persistent courses of conduct, and/or deriving substantial revenue from goods and services provided to individuals in Delaware and in this judicial district.

7. Venue is proper in this judicial district pursuant to 28 U.S.C. § 1400(b) because Defendant resides in this District.

FACTUAL ALLEGATIONS

8. On January 15, 2002, the United States Patent and Trademark Office ("USPTO") duly and legally issued the '769 patent, entitled "Query optimization by transparently altering properties of relational tables using materialized views" after a full and fair examination. (Exhibit A).

9. Devine is presently the owner of the patent, having received all right, title and interest in and to the '769 patent from the previous assignee of record. Devine possesses all rights of recovery under the '769 patent, including the exclusive right to recover for past infringement.

10. The '769 patent contains six independent claims and sixty-six dependent claims. Defendant commercializes, inter alia, methods that perform all the steps recited in at least one claim of the '769 patent.

11. The invention claimed in the '769 patent comprises a method optimizing database queries using a materialized view for a table referenced in the query, wherein the materialized view has different properties than the referenced table.

12. The method allows a user to optimize a query in a computer system by transparently altering properties of relational tables using materialized views.

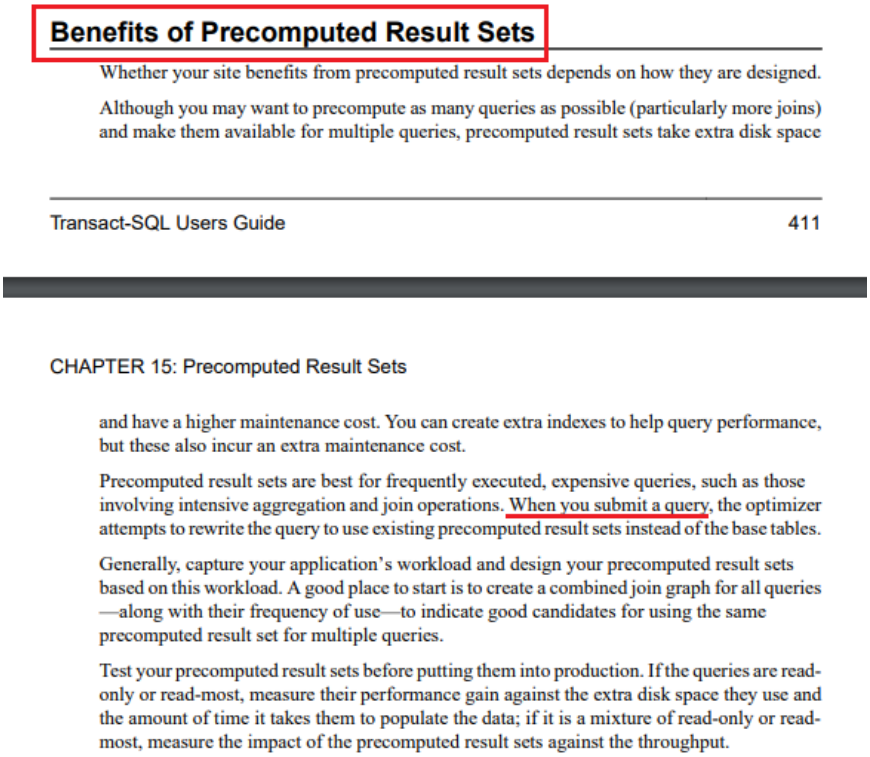
13. The technology embodied by the '769 patent improved techniques for the replication of materialized views in a massively parallel processing (MPP) environment.

DEFENDANT'S PRODUCTS

14. Defendant offers products, such as the "SAP Adaptive Enterprise Server 16.0" (the "Accused Instrumentality"), that practices a method of optimizing a query (e.g., by means of a query optimizer) in a computer system, the query being performed by the computer system to retrieve data from a database stored on the computer system, as recited in the preamble of claim 1 of the '769 patent and as shown on Defendant's website[1].

15. As recited in the first step of claim 1, the Accused Instrumentality practices accepting the query into the computer system by allowing a user to submit a query[2].

[1] http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.help.ase.16.0/doc/html/title.html, last visited April 12, 2019.
[2] http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc32300.1600/doc/pdf/ase_tsql.pdf, last visited April 12, 2019.



Benefits of Precomputed Result Sets

Whether your site benefits from precomputed result sets depends on how they are designed.

Although you may want to precompute as many queries as possible (particularly more joins) and make them available for multiple queries, precomputed result sets take extra disk space

Transact-SQL Users Guide 411

CHAPTER 15: Precomputed Result Sets

and have a higher maintenance cost. You can create extra indexes to help query performance, but these also incur an extra maintenance cost.

Precomputed result sets are best for frequently executed, expensive queries, such as those involving intensive aggregation and join operations. When you submit a query, the optimizer attempts to rewrite the query to use existing precomputed result sets instead of the base tables.

Generally, capture your application's workload and design your precomputed result sets based on this workload. A good place to start is to create a combined join graph for all queries —along with their frequency of use—to indicate good candidates for using the same precomputed result set for multiple queries.

Test your precomputed result sets before putting them into production. If the queries are read-only or read-most, measure their performance gain against the extra disk space they use and the amount of time it takes them to populate the data; if it is a mixture of read-only or read-most, measure the impact of the precomputed result sets against the throughput.

16. As recited in the second step of claim 1, the Accused Instrumentality practices determining whether there exist one or more materialized views for one or more tables referenced in the query, wherein the materialized view has different partitioning or replication properties than the tables referenced in the query. For example, the Accused Instrumentality uses precomputed result sets, which can be partitioned or its other properties can be changed independently and differently vis a vis the actual table referenced by a query.[3]

[3] http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc32300.1600/doc/pdf/ase_tsql.pdf, last visited April 12, 2019.

Altering Precomputed Result Sets

Use the **alter** command to change the precomputed result set's policies or properties.

The syntax is:

```
alter {precomputed result set | materialized view}
prs_name
{immediate | manual} refresh
```



```
| enable | disable
| {enable | disable} use in optimization
```

This example alters the `author_prs` precomputed result set from **manual** to **immediate**:

```
alter precomputed result set author_prs
immediate refresh
```

Next

alter automatically refreshes the precomputed result set when you change from a **manual** to an **immediate** refresh, or from **disable** to **enable**. Altering a precomputed result set for **disable use in optimization** prevents the precomputed result set from participating in future query rewriting. However, any plans already cached using the precomputed result set are not recompiled.

alter table

Makes changes to existing tables.

- Adds new columns to a table; drops or modifies existing columns; adds, changes, or drops constraints; changes properties of an existing table; enables or disables triggers on a table, changes the compression level of a table.
- Supports adding, dropping, and modifying computed columns, and enables the materialized property, nullability, or definition of an existing computed column to be changed.
- Partitions and repartitions a table with specified partition strategy, adds partitions to a table with existing partitions, and splits or merges existing partitions.

Syntax

```
alter table [[database.][owner].]table_name
    {add column_name datatype}
        [default {constant_expression | user | null}]
        {identity | null | not null [not materialized]}
        [off row | in row]
        [[constraint constraint_name]
        {{unique | primary key}
            [clustered | nonclustered]
            [asc | desc]
            [with {fillfactor = pct,
                max_rows_per_page = num_rows,
                reservepagegap = num_pages
                immediate_allocation]
            [on segment_name]
        | references [[database.][owner].]ref_table
            [(ref_column)]
            [match full]
        | check (search_condition)]
        [encrypt [with [[database.][owner].] keyname]
            [decrypt_default {constant_expression | null}]]
        [compressed = compression_level | not compressed]
        [, next_column]...
```

17. As recited in the third step of claim 1, the Accused Instrumentality practices analyzing whether at least a portion of the query can be evaluated using one or more of the

materialized views in a local fashion, so that no data movement is required for the evaluation. For example, the Accused Instrumentality uses a query optimizer to attempt to rewrite a query using precomputed results sets.

CHAPTER 15 **Precomputed Result Sets**

A precomputed result set is a view for which the result is computed, stored, and available for future use. Once configured for precomputed result sets, SAP ASE precomputes queries and attempts to use the precomputed result during subsequent iterations. Precomputed result sets are also called materialized views.

Conceptually, a precomputed result set is both a view (because it includes query definition stored in the system tables) and a table (because it includes persistent data). You can perform many of the same operations that you perform on tables on precomputed result sets, including creating indexes and running update statistics.

Once SAP ASE is configured to use precomputed result sets, the optimizer attempts to automatically rewrite each query using a precomputed result set. However, the final plan the optimizer selects is primarily cost based.

When the optimizer rewrites a query using a precomputed result set, it decides which precomputed result set is the best candidate. If the optimizer chooses to replace all, or part, of a query with a precomputed result set, it also adds any necessary compensation to the rewritten query (that is, any predicates needed to ensure the rewritten query is equivalent to the original user query). For example, if the user query includes a join of:

```
c1=c2 and c2=c3 and c3=c4
```

but the precomputed result set includes a join for:

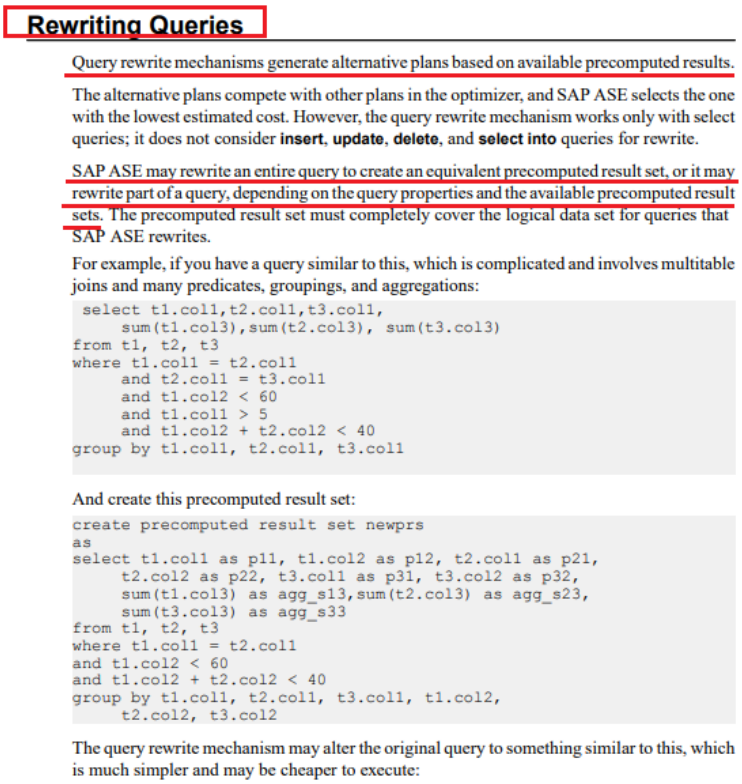
```
c1=c2 and c3=c4
```

the rewritten query using the precomputed result set must have a compensation predicate similar to c1=c3 to form an equivalent query.

Like an index, a precomputed result set has a maintenance cost for concurrent **insert**, **update**, and **delete** statements. Generally, precomputed result-set maintenance overhead consists of more than maintaining the indexes when the definition involves multiple table joins. Consequently, precomputed result sets are unsuitable for OLTP with heavy concurrent **insert**, **update**, and **delete** statements and simple index-based **select** statements.

Note: SAP ASE allows you to run **updates statistics** on precomputed result sets.

18. As recited in the fourth step of claim 1, the Accused Instrumentality practices rewriting the query to use one or more materialized views rather than an original table or tables referenced in the query. For example, the Accused Instrumentality rewrites queries using precomputed result sets when possible.



Rewriting Queries

Query rewrite mechanisms generate alternative plans based on available precomputed results.

The alternative plans compete with other plans in the optimizer, and SAP ASE selects the one with the lowest estimated cost. However, the query rewrite mechanism works only with select queries; it does not consider **insert**, **update**, **delete**, and **select into** queries for rewrite.

SAP ASE may rewrite an entire query to create an equivalent precomputed result set, or it may rewrite part of a query, depending on the query properties and the available precomputed result sets. The precomputed result set must completely cover the logical data set for queries that SAP ASE rewrites.

For example, if you have a query similar to this, which is complicated and involves multitable joins and many predicates, groupings, and aggregations:

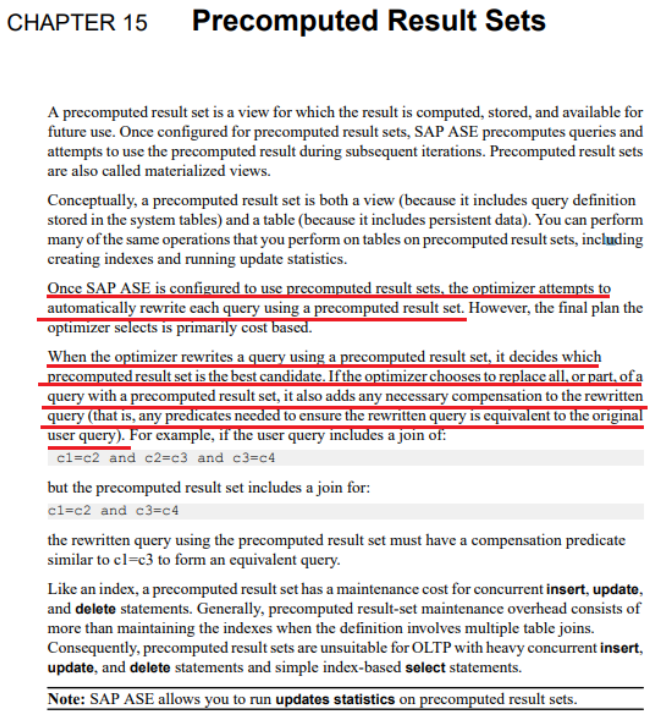
```
select t1.col1,t2.col1,t3.col1,
     sum(t1.col3),sum(t2.col3), sum(t3.col3)
from t1, t2, t3
where t1.col1 = t2.col1
     and t2.col1 = t3.col1
     and t1.col2 < 60
     and t1.col1 > 5
     and t1.col2 + t2.col2 < 40
group by t1.col1, t2.col1, t3.col1
```

And create this precomputed result set:

```
create precomputed result set newprs
as
select t1.col1 as p11, t1.col2 as p12, t2.col1 as p21,
     t2.col2 as p22, t3.col1 as p31, t3.col2 as p32,
     sum(t1.col3) as agg_s13,sum(t2.col3) as agg_s23,
     sum(t3.col3) as agg_s33
from t1, t2, t3
where t1.col1 = t2.col1
and t1.col2 < 60
and t1.col2 + t2.col2 < 40
group by t1.col1, t2.col1, t3.col1, t1.col2,
     t2.col2, t3.col2
```

The query rewrite mechanism may alter the original query to something similar to this, which is much simpler and may be cheaper to execute:

19. As recited in the fifth step of claim 1, the Accused Instrumentality practices executing the rewritten query using one or more materialized views. For example, the Accused Instrumentality replaces all or part of a query with a precomputed result set and adds any necessary compensation to ensure the rewritten query is executed as the original.



CHAPTER 15 **Precomputed Result Sets**

A precomputed result set is a view for which the result is computed, stored, and available for future use. Once configured for precomputed result sets, SAP ASE precomputes queries and attempts to use the precomputed result during subsequent iterations. Precomputed result sets are also called materialized views.

Conceptually, a precomputed result set is both a view (because it includes query definition stored in the system tables) and a table (because it includes persistent data). You can perform many of the same operations that you perform on tables on precomputed result sets, including creating indexes and running update statistics.

Once SAP ASE is configured to use precomputed result sets, the optimizer attempts to automatically rewrite each query using a precomputed result set. However, the final plan the optimizer selects is primarily cost based.

When the optimizer rewrites a query using a precomputed result set, it decides which precomputed result set is the best candidate. If the optimizer chooses to replace all, or part, of a query with a precomputed result set, it also adds any necessary compensation to the rewritten query (that is, any predicates needed to ensure the rewritten query is equivalent to the original user query). For example, if the user query includes a join of:

```
c1=c2 and c2=c3 and c3=c4
```

but the precomputed result set includes a join for:

```
c1=c2 and c3=c4
```

the rewritten query using the precomputed result set must have a compensation predicate similar to c1=c3 to form an equivalent query.

Like an index, a precomputed result set has a maintenance cost for concurrent **insert**, **update**, and **delete** statements. Generally, precomputed result-set maintenance overhead consists of more than maintaining the indexes when the definition involves multiple table joins. Consequently, precomputed result sets are unsuitable for OLTP with heavy concurrent **insert**, **update**, and **delete** statements and simple index-based **select** statements.

Note: SAP ASE allows you to run **updates statistics** on precomputed result sets.

20. The elements described in paragraphs 14-19 are covered by at least claim 1 of the '769 patent. Thus, Defendant's use of the Accused Product is enabled by the method described in the '769 patent.

COUNT I
(DIRECT INFRINGEMENT OF THE '769 PATENT)

21. Plaintiff realleges and incorporates by reference the allegations set forth in paragraphs 1 to 20.

22. Defendant has, prior to launching the Accused Product in the United States, performed internal testing with said Accused Product.

23. In violation of 35 U.S.C. § 271, Defendant is now, and has been directly infringing the '769 patent.

24. Defendant has had knowledge of infringement of the '769 patent at least as of the service of the present complaint.

25. Defendant has directly infringed and continues to directly infringe at least claim 1 of the '769 patent by using, at least through internal testing, the Accused Instrumentality without authority in the United States, and will continue to do so unless enjoined by this Court. As a direct and proximate result of Defendant's direct infringement of the '769 patent, Plaintiff has been and continues to be damaged.

26. By engaging in the conduct described herein, Defendant has injured Devine and is thus liable for infringement of the '769 patent, pursuant to 35 U.S.C. § 271.

27. Defendant has committed these acts of infringement without license or authorization.

28. As a result of Defendant's infringement of the '769 patent, Devine has suffered monetary damages and is entitled to a monetary judgment in an amount adequate to compensate for Defendant's past infringement, together with interests and costs.

COUNT II
(INDIRECT INFRINGEMENT OF THE '769 PATENT)

29. Plaintiff realleges and incorporates by reference the allegations set forth in paragraphs 1 to 28.

30. In violation of 35 U.S.C. § 271, Defendant is now, and has been indirectly infringing the '769 patent.

31. Defendant has had knowledge of infringement of the '769 patent at least as of the service of the present complaint.

32. Defendant has indirectly infringed and continues to indirectly infringe at least claim 1 of the '769 patent by actively inducing its respective customers, users, and/or licensees to directly infringe by using the Accused product. Defendant engaged or will have engaged in such inducement having knowledge of the '769 patent. Furthermore, Defendant knew or should have

known that its action would induce direct infringement by others and intended that its actions would induce direct infringement by others. For example, Defendant sells, offers to sell and advertises the Accused Product through websites or digital distribution platforms that are available in Delaware, specifically intending that its customers use it. Furthermore, Defendant's customers' use of the Accused Product is facilitated by the invention described in the '769 patent. As a direct and proximate result of Defendant's indirect infringement by inducement of the '769 patent, Plaintiff has been and continues to be damaged.

33. By engaging in the conduct described herein, Defendant has injured Devine and is thus liable for infringement of the '769 patent, pursuant to 35 U.S.C. § 271.

34. Defendant has committed these acts of infringement without license or authorization.

35. As a result of Defendant's infringement of the '769 patent, Devine has suffered monetary damages and is entitled to a monetary judgment in an amount adequate to compensate for Defendant's past infringement, together with interests and costs.

DEMAND FOR JURY TRIAL

36. Devine demands a trial by jury of any and all causes of action.

PRAYER FOR RELIEF

WHEREFORE, Devine prays for the following relief:

a. That Defendant be adjudged to have directly infringed the Patents-In-Suit either literally or under the doctrine of equivalents;

b. An accounting of all infringing sales including, but not limited to, those sales not presented at trial;

c. That Defendant, its officers, directors, agents, servants, employees, attorneys, affiliates, divisions, branches, parents, and those persons in active concert or participation with any of them, be permanently restrained and enjoined from directly infringing the Patent-In-Suit;

d. An award of damages pursuant to 35 U.S.C. § 284 sufficient to compensate Devine for the Defendant's past infringement, including compensatory damages;

e. An assessment of pre-judgment and post-judgment interest and costs against Defendant, together with an award of such interest and costs, in accordance with 35 U.S.C. § 284; and

f. That Devine have such other and further relief as this Court may deem just and proper.

Dated: April 29, 2019

DEVLIN LAW FIRM LLC

/s/ Timothy Devlin

Timothy Devlin (No. 4241)
1306 N. Broom Street, 1st Floor
Wilmington, DE 19806
Telephone: (302) 449-9010
Facsimile: (302) 353-4251
Email: tdevlin@devlinlawfirm.com

Eugenio J. Torres-Oyola
USDC No. 215505
Jean G. Vidal Font
USDC No. 227811
(*Pro Hac Vice* Applications Pending)
FERRAIUOLI LLC
221 Plaza, 5th Floor
221 Ponce de León Avenue
San Juan, PR 00917
Telephone: (787) 766-7000
Facsimile: (787) 766-7001
Email: etorres@ferraiuoli.com
Email: jvidal@ferraiuoli.com

ATTORNEYS FOR PLAINTIFF DEVINE LICENSING LLC