Modelo de reconocimiento de variables que influyen en el rendimiento de transacciones de RDBMS aplicando PCA y PCR

José Luis Ponce Vergara
ponce_joseluis@yahoo.com
Universidad Nacional de Ingeniería, Lima, Perú

Recibido: 31/5/2021 Aceptado: 23/9/2021
doi: https://doi.org/10.26439/ciis2021.5581

RESUMEN. El conocimiento de los factores que influencian la eficiencia de un sistema es importante para su correcta administración y mantenimiento. Múltiples organizaciones soportan sus operaciones con aplicaciones que interactúan con un sistema de gestión de bases de datos relacional (RDBMS), las cuales pueden mejorar su eficiencia a través del conocimiento de los factores que influyen en el rendimiento de las ejecuciones de las sentencias SQL que conforman sus cargas de trabajo, especialmente las cargas de trabajo generadas por aplicaciones implementadas en ambientes de producción que manifiestan recurrencia en el tiempo. El artículo de investigación que se presenta propone un modelo de reconocimiento de factores que afectan el rendimiento de las ejecuciones de las sentencias SQL que se procesan en un RDBMS y discute la implementación de una técnica de predicción de métrica de rendimiento, valiéndose de los algoritmos de aprendizaje de máquina estadístico denominados Análisis de Componentes Principales (PCA) y Regresión de Componentes Principales (PCR), que explotan la información de los planes, estadísticas y métricas generadas durante el ciclo de vida de las ejecuciones de las sentencias SQL.

PALABRAS CLAVE: RDBMS / algoritmo de aprendizaje de máquina estadístico / PCA / PCR / rendimiento de ejecución de sentencias SQL / administración del rendimiento

RECOGNITION MODEL OF VARIABLES THAT INFLUENCE THE PERFORMANCE OF RDBMS TRANSACTIONS APPLYING PCA AND PCR

ABSTRACT. Knowledge of the factors that influence the efficiency of a system is essential for its administration and maintenance. Likewise, various organizations support their operations with applications that interact with a Relational Database Management System (RDBMS), which can improve their efficiency through knowledge of the factors that influence the performance of SQL statement executions. That makes up the workload, especially the workloads generated by applications implemented in Production environments that show recurrence over time. The present research article proposes a model for the recognition of factors that affect the performance of the executions of the SQL statements that are processed in an RDBMS and discusses the implementation of a performance metric prediction technique, using algorithms of statistical machine learning called Principal Component Analysis (PCA) and Principal Components Regression (PCR), which exploit the information of the plans, statistics, and metrics generated during the life cycle of the executions of the SQL sentences.

KEYWORDS: RDBMS / Statistical Machine Learning Algorithm / Principal Component Analysis, PCA / Principal Components Regression, PCR / SQL Performanc / Performance Management

1. INTRODUCCIÓN

El artículo de investigación tiene como objetivo generar un modelo que permita predecir la respuesta de un sistema de gestión de base de datos relacional (RDBMS), con una carga de trabajo conocida o recurrente, a través del conocimiento de los factores que influyen en el rendimiento de las ejecuciones de las sentencias SQL que se ejecutan en tal sistema. El contexto del artículo se enmarca en dos áreas de conocimientos conocidas como "ingeniería de características o atributos" (*feature engineering*) e "ingeniería de rendimiento de *software*" (*software performance engineering*). La ingeniería de características es la que selecciona los atributos más relevantes de conjuntos de datos considerables y que según Lam *et al.* (2017) es una tarea que requiere personal experto en esta área con amplio conocimiento del tema que se investiga. La ingeniería de rendimiento de *software* representa a las actividades dirigidas a cumplir los requerimientos de rendimiento, según Woodside *et al*. (2007).

Tener identificados los factores que gobiernan el rendimiento, y a partir de allí generar una estimación de rendimiento, permitirá gestionar de manera eficiente las operaciones que el RDBMS ejecuta a través del tiempo. Entiéndase al rendimiento (*performance*) de un sistema de información como un requerimiento no funcional (NFR, *non-funtional requirement*) que se debe satisfacer para brindar un adecuado nivel de servicio.

El modelo planteado no influye ni condiciona *a priori* la determinación de los factores a considerar, y delega a los algoritmos de aprendizaje de máquina esa decisión. Cabe resaltar que se ha obviado información subyacente, que podría ser incluida en futuras investigaciones como, por ejemplo, características particulares del *hardware* del servidor de base de datos que aloja al RDBMS, pero que no desvirtúan los resultados obtenidos. De la manera planteada, el modelo es portable a un universo de plataformas mucho mayor.

La investigación propone un modelo que se fundamenta sobre las investigaciones previas realizadas en las siguientes dos áreas del conocimiento:

a. Reducción de la dimensionalidad de un conjunto de datos para la identificación de los factores que influyen en el rendimiento de las ejecuciones de las sentencias SQL.

   El objetivo de este esfuerzo es reducir el número de variables de un conjunto de datos observados sin perder su esencia numérica (medidas de tendencia y varianza). El modelo propuesto utiliza el algoritmo de aprendizaje de máquina estadístico Análisis de Componentes Principales (*Principal Component Analysis* [PCA]), que es una técnica de aprendizaje no supervisado reconocida para estos menesteres, lo cual se fundamenta en la evaluación hecha por Mikolajczyk y Schmid (2005). Asimismo, está probada su justificación y uso debido a la existencia de investigaciones que apoyan sus soluciones propuestas empleando el algoritmo PCA, entre las cuales se puede citar a Badrinath *et al.* (2015), Brauckhoff *et al.* (2015), Shyu *et al.* (2003) y Shawe-Taylor (2004).

b. Predicción de respuesta del sistema sobre la base de componentes principales, utilizando el algoritmo Regresión de Componentes Principales (*Principal Components Regression* [PCR]), que utiliza los primeros "M" componentes principales, y usa estos componentes como variables independientes en un modelo de regresión lineal ajustado por mínimos cuadrados.

   La idea clave es que generalmente un número reducido de componentes principales es suficiente para explicar la variabilidad de los datos, así como la relación con la respuesta o salida del sistema (variable dependiente) que se desea evaluar. Adicionalmente, con esta técnica se mitiga el problema del sobreajuste (*overfitting*) (James *et al*., 2013; Lee et al., 2015; Hadi y Ling, 1998).

2. ESTADO DEL ARTE

2.1 Estado del arte de la investigación sobre análisis de rendimiento de *software*

Las investigaciones relativas al tema se pueden clasificar bajo los siguientes tres enfoques.

- *Enfoque analítico.* Donde se modela al sistema de *software* como un conjunto de entidades servidoras dispuestas de diferentes maneras, que procesan tareas en un orden de llegada preestablecido. Abordan este enfoque Kleinrock (1976), así como Smith y Williams (2000). También se incluyen bajo este enfoque los modelos abstractos que con base en fórmulas y algoritmos reproducen los flujos reales de procesamiento e implementan cargas de trabajo simuladas (Duggan *et al.*, 2011).
- *Enfoque estadístico o de aprendizaje de máquina estadístico.* Basado y motivado por la existencia de datos relativos al tema que se investiga (*data-driven*), y algoritmos de aprendizaje de máquina. Este enfoque es el utilizado en este artículo. Bajo este enfoque se aprovechan las técnicas de análisis de regresión estadística, econométricas o algoritmos de aprendizaje de máquina, con un conjunto de variables de entrada y otro conjunto de datos con una o más variables de salida, las cuales se consideran dependientes de las entradas. Los conjuntos de observaciones para entrenar (*training*) y probar (*testing*) los modelos son finitos, siendo el primero de mayor cardinalidad que el segundo. El modelo resultante es un modelo predictivo, entrenado y puesto a prueba con observaciones previas, y que permite estimar los valores de las variables de salida ante ciertos valores asignados a las variables de entrada. Abordan este enfoque Bontempi y Kruijtzer (2002), Giusto *et al.* (2001), Lam *et al.* (2017).
- *Enfoque de simulación de sistemas.* Basado en la construcción de un modelo de simulación que captura las partes que marcan la cadencia del flujo de procesamiento del sistema de información que se estudia. El modelo de simulación imita el comportamiento en

tiempo de ejecución del sistema, con el objetivo de ganar conocimiento del sistema e identificar los cuellos de botella de este. Se puede consultar la investigación de Fortier y Michel (2003) para contar con una evaluación detallada de productos orientados bajo este enfoque.

2.2 Investigaciones previas en el campo de estudio

Para obtener una caracterización de la carga de trabajo SQL, Yu *et al.* (1992) utilizan un sistema *ad hoc* (REDWAR) que perfila sus estadísticas, indicadores, singularidades y métricas. Esta caracterización sirve como base para evaluaciones de tipo *benchmarking*. En la misma línea, Panda *et al.* (2015) amplían la investigación incluyendo bases de datos NoSQL.

De *et al.* (2001) realizan una investigación empírica de factores influyentes sobre el rendimiento de consultas en bases de datos orientadas a objetos, donde se concentran en la naturaleza de las consultas, sintaxis y semántica, así como en los modelos de datos de implementación. Este tipo de enfoques requiere revisar cuestiones de diseño y programación de un sistema de información, tales como el código de las consultas y el tipo de modelo de datos elegido, lo cual impide escalar con facilidad la propuesta a otras situaciones. Asimismo, Schkolnick y Tiberio (1985) plantean el desarrollo de fórmulas de costo para actualizaciones de bases de datos relacionales, como resultado de la evaluación de la sentencia a ejecutar, donde se consideran como variables de entrada a los tipos de datos y predicados utilizados como filtros. El autor propone que este costo calculado influye considerablemente tanto en el diseño del modelo de datos como en las rutas de acceso a los datos. Por otro lado, Ganapathi *et al.* (2009, 2015) plantean un procedimiento para predecir métricas de rendimiento, donde se incluye el tiempo de respuesta de una sentencia SQL y se toma como entrada los planes de ejecución de las sentencias SQL, así como sus resultados, para generar dos matrices: *Query Plan Feature Matrix* y *Performance Feature Matrix*. A partir de dichas matrices se generan proyecciones de rendimiento utilizando un motor de aprendizaje de máquina.

Como resultado de esta revisión, se observa que hay diferencias metodológicas al tratar de encontrar los factores de influencia en el rendimiento de las operaciones de un sistema de gestión de base de datos. Algunos enfoques analizan las sentencias SQL que conforman la carga de trabajo y sobre este análisis generan conclusiones (análisis de caja blanca), pero son más difíciles de generalizar por el trabajo que conllevan. Otros enfoques generan predicciones de rendimiento sobre la base de factores establecidos, y utilizan estos valores y los resultados obtenidos en las ejecuciones para entrenar y calibrar algoritmos de aprendizaje de máquina (análisis de caja negra). La investigación que se presenta sigue el enfoque de caja negra, por permitir generalizar su metodología a diversas situaciones.

El artículo presenta en la sección 3 la metodología que se sigue para generar el modelo de identificación de factores y predicción de rendimiento; luego, en la sección 4 se muestran

los resultados obtenidos por la aplicación del modelo en casuística real de ambientes de producción, y culmina con la sección 5 donde se presentan las conclusiones obtenidas en esta investigación.

3. METODOLOGÍA

La metodología toma como insumo las muestras de información relacionadas con la interpretación y ejecución de las sentencias SQL clasificadas en dos grupos:

- Grupo 1: datos previos a la ejecución de las sentencias SQL, relacionados con el análisis e interpretación de las sentencias SQL. Son las variables de tipo numérico contenidas en el historial de planes de ejecución de las sentencias SQL. Esta información es generada por el optimizador del motor de la base de datos antes de ejecutar las sentencias SQL (véase la tabla 1).

Tabla 1
Datos previos a la ejecución de las sentencias SQL

Columna	Tipo de dato	Nulidad	Descripción
DBID	NUMBER	NOT NULL	Database ID
PLAN_HASH_VALUE	NUMBER	NOT NULL	Numerical representation of the SQL plan for the cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line).
ID	NUMBER	NOT NULL	A number assigned to each step in the execution plan
OBJECT#	NUMBER		Object number of the table or the index
PARENT_ID	NUMBER		ID of the next execution step that operates on the output of the current step
DEPTH	NUMBER		Depth (or level) of the operation in the tree. It is not necessary to issue a CONNECT BY statement to get the level information, which is generally used to indent the rows from the PLAN_TABLE table. The root operation (statement) is level 0.
POSITION	NUMBER		Order of processing for all operations that have the same PARENT_ID
SEARCH_COLUMNS	NUMBER		Number of index columns with start and stop keys (that is, the number of columns with matching predicates)

(*continúa*)

(*continuación*

COST	NUMBER	Cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null.
CARDINALITY	NUMBER	Estimate, by the cost-based optimizer, of the number of rows produced by the operation
BYTES	NUMBER	Estimate, by the cost-based optimizer, of the number of bytes produced by the operation
PARTITION_ID	NUMBER	Step that computes the pair of values of the PARTITION_START and PARTITION_STOP columns
CPU_COST	NUMBER	CPU cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null.
IO_COST	NUMBER	I/O cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null.
TEMP_SPACE	NUMBER	Temporary space usage of the operation (sort or hash-join) as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null.
TIME	NUMBER	Elapsed time (in seconds) of the operation as estimated by the optimizer's cost-based approach. For statements that use the rulebased approach, this column is null.

Elaboración propia

- Grupo 2: datos posteriores a la ejecución de las sentencias SQL. Son las variables de tipo numérico contenidas en el historial de estadísticas y métricas obtenidas luego de la ejecución de las sentencias SQL. Este grupo permitirá entrenar y evaluar al algoritmo de PCR para la predicción del valor deseado de rendimiento. En esta investigación se escoge el tiempo de respuesta (*elapsed time*) (véase la tabla 2).

Tabla 2
Datos posteriores a la ejecución de las sentencias SQL

COLUMNA	TIPO DE DATO	NULIDAD	DESCRIPCIÓN
SNAP_ID	NUMBER	NOT NULL	Unique snapshot ID
DBID	NUMBER	NOT NULL	Database ID for the snapshot
INSTANCE_NUMBER	NUMBER	NOT NULL	Instance number for the snapshot
PLAN_HASH_VALUE	NUMBER	NOT NULL	Numerical representation of the SQL plan for the cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line).
OPTIMIZER_COST	NUMBER		Cost of the query given by the optimizer
OPTIMIZER_ENV_HASH_VALUE	NUMBER		Hash Value for the optimizer environment
SHARABLE_MEM	NUMBER		Amount of shared memory used by the child cursor (in bytes)
LOADED_VERSIONS	NUMBER		Indicates whether the context heap is loaded (1) or not (0)
VERSION_COUNT	NUMBER		Number of children associated with the cursor
FORCE_MATCHING_SIGNATURE	NUMBER		The signature used when the CURSOR_SHARING parameter is set to FORCE
FETCHES_DELTA	NUMBER		Delta number of fetches associated with the SQL statement
END_OF_FETCH_COUNT_DELTA	NUMBER		Delta number of times this cursor was fully executed since the cursor was brought into the library cache. The value of this statistic is not incremented when the cursor is partially executed, either because it failed during the execution or because only the first few rows produced by this cursor are fetched before the cursor is closed or re-executed.
SORTS_DELTA	NUMBER		Delta number of sorts that were done for this child cursor
EXECUTIONS_DELTA	NUMBER		Delta number of executions that took place on this object since it was brought into the library cache
PX_SERVERS_EXECS_DELTA	NUMBER		Delta number of PX server executions
LOADS_DELTA	NUMBER		Delta number of times the object was either loaded or reloaded

(*continúa*)

(*continuación*)

INVALIDATIONS_DELTA	NUMBER	Delta number of times this child cursor has been invalidated
PARSE_CALLS_DELTA	NUMBER	Delta number of parse calls for this child cursor
DISK_READS_DELTA	NUMBER	Delta number of disk reads for this child cursor
BUFFER_GETS_DELTA	NUMBER	Delta number of buffer gets for this child cursor
ROWS_PROCESSED_DELTA	NUMBER	Delta number of rows the parsed SQL statement returns
CPU_TIME_DELTA	NUMBER	Delta value of CPU time (in microseconds) used by this cursor for parsing/executing/fetching
ELAPSED_TIME_DELTA	NUMBER	Delta value of elapsed time (in microseconds) used by this cursor for parsing/executing/fetching
IOWAIT_DELTA	NUMBER	Delta value of user I/O wait time (in microseconds)
CLWAIT_DELTA	NUMBER	Delta value of cluster wait time (in microseconds)
APWAIT_DELTA	NUMBER	Delta value of application wait time (in microseconds)
CCWAIT_DELTA	NUMBER	Delta value of concurrency wait time (in microseconds)
DIRECT_WRITES_DELTA	NUMBER	Delta value of direct writes
PLSEXEC_TIME_DELTA	NUMBER	Delta value of PL/SQL Execution Time (in microseconds)
JAVEXEC_TIME_DELTA	NUMBER	Delta value of Java Execution Time (in microseconds)
IO_INTERCONNECT_BYTES_DELTA	NUMBER	Delta value of number of I/O bytes exchanged between Oracle Database and the storage system
PHYSICAL_READ_REQUESTS_DELTA	NUMBER	Delta value of number of physical read I/O requests issued by the monitored SQL
PHYSICAL_READ_BYTES_DELTA	NUMBER	Delta value of number of bytes read from disks by the monitored SQL
PHYSICAL_WRITE_REQUESTS_DELTA	NUMBER	Delta value of number of physical write I/O requests issued by the monitored SQL
PHYSICAL_WRITE_BYTES_DELTA	NUMBER	Delta value of number of bytes written to disks by the monitored SQL

(*continúa*)

(*continuación*)

OPTIMIZED_PHYSICAL_READS_DELTA	NUMBER	Delta value of number of physical reads from Database Smart Flash Cache by the monitored SQL

Elaboración propia

Para dar inicio a la identificación de los factores que influyen en el rendimiento de las ejecuciones de las sentencias SQL, se preparan los datos por utilizar (Anderson y Cafarella, 2016). Para ello, se seleccionan las variables numéricas representativas antes de la ejecución de las sentencias SQL. Las variables pertinentes al momento antes de la ejecución de las sentencias SQL se extraen desde el historial de los planes de ejecución generados por el optimizador de base de datos relacional y se agregan (sintetizan), debido a que el plan de ejecución de una sentencia SQL tiene la forma de árbol invertido. Téngase en cuenta que tanto los planes de ejecución como las estadísticas y métricas son grabadas por el motor de base de datos relacional en un repositorio dedicado a ese fin y dicha información es considerada como insumo del modelo propuesto.

Las variables obtenidas son reducidas dimensionalmente (simplificadas) a través de un algoritmo de aprendizaje de máquina estadístico de reducción de dimensionalidad llamado Análisis de Componentes Principales (*Principal Component Analysis* [PCA]), para posteriormente a través del algoritmo de Regresión de Componentes Principales (*Principal Components Regression*[PCR]) obtener la función de predicción del tiempo de respuesta de la ejecución de una sentencia SQL.

3.1 Lineamientos del diseño de experimentos

Se ha tenido en cuenta para las pruebas de campo el principio de aleatorización, necesario para asignar eficientemente las combinaciones de los factores a las ejecuciones de sentencias SQL (unidades experimentales). Para la aleatorización, se han definido muestras de rendimiento al azar de bases de datos de propósito definido, correspondientes a diferentes situaciones y momentos de sus cargas de trabajo. Se intenta de este modo prevenir la introducción de sesgos en el experimento y evitar la dependencia entre observaciones.

3.2 Entorno de implementación del modelo

El diseño, la codificación y las pruebas del modelo se implementaron utilizando como insumos los resultados de rendimiento pertenecientes a bases de datos relacionales Oracle Database 12c Release 1 Enterprise Edition que operan en entornos de producción (véase la tabla 3). Las sentencias SQL que conforman las cargas de trabajo son sentencias DML (INSERT, UPDATE, DELETE, MERGE) y consultas (SELECT). Se desarrollaron herramientas ETL

para preparar y adecuar estos datos al propósito buscado. La información de rendimiento de las ejecuciones de sentencias SQL es filtrada, procesada, agregada y cargada a un modelo de datos *ad hoc*, para luego ser explotada por los algoritmos de aprendizaje de máquina estadístico PCA y PCR. Estas herramientas se programaron a través de *shell scripts* y SQL *scripts*. Los algoritmos de aprendizaje de máquina estadístico se programaron en lenguaje R versión 3.6.1 para plataforma Windows 64 *bits*.

Tabla 3
Bases de datos en estudio

Base de datos	Propósito (carga de trabajo)	Tipo de base de datos
1	Gestión financiera (Batch + DSS)	DSS
2	Operaciones (transaccional)	OLTP
3	MRP (transaccional y Batch + DSS)	OLTP + DSS

Elaboración propia

El modelo planteado en su forma general se puede visualizar en la figura 1. En la figura 2 se muestra el proceso de reducción de la dimensionalidad del conjunto agregado de datos previos a la ejecución de las sentencias SQL utilizando un algoritmo basado en PCA. En la figura 3 se muestra el proceso de predicción del tiempo de respuesta de sentencias SQL utilizando el algoritmo denominado *Principal Components Regression*.

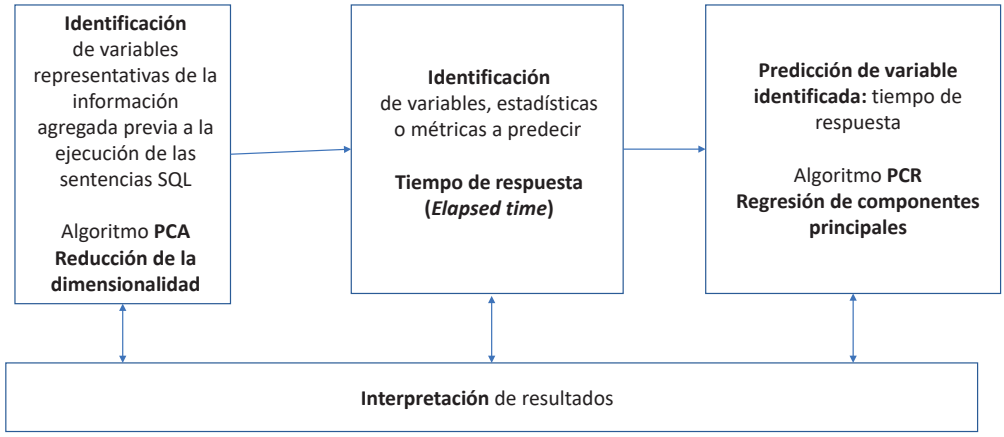

Figura 1. Método de síntesis (reducción de dimensionalidad) y regresión de componentes principales sobre variables que afectan el rendimiento de las ejecuciones de sentencias SQL
Elaboración propia

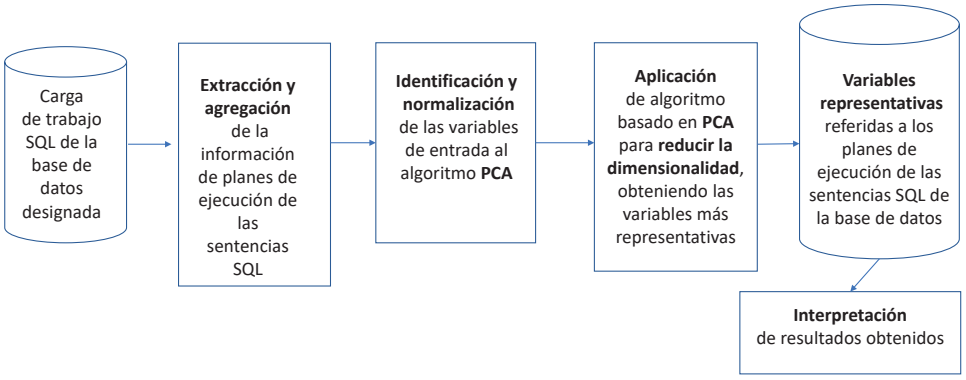

Figura 2. Reducción de la dimensionalidad del conjunto agregado de datos de planes de ejecución
Elaboración propia

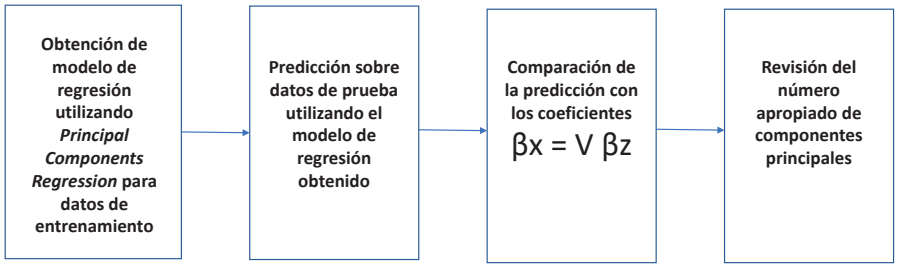

Figura 3. Reducción de la dimensionalidad del conjunto agregado de datos referidos al historial estadístico de ejecuciones
Elaboración propia

4. RESULTADOS

4.1 Aplicación de algoritmo PCA a la información previa a la ejecución de sentencias SQL

Para bases de datos de producción que sirven a aplicaciones DSS y aplicando los algoritmos de reducción de dimensionalidad al conjunto agregado de datos previos a la ejecución de las sentencias SQL, se identifican los siguientes factores en el Componente Principal 1 (PC1), que es donde se concentra la mayor parte de varianza del conjunto de datos: I/O Cost, Cost, CPU Cost, Bytes y Cardinality (véase la figura 4). De la misma manera, para bases de datos OLTP se identifican los factores I/O Cost, Cost, CPU Cost, Bytes y Cardinality (véase la figura 5). Análogamente, para bases de datos con carga de trabajo mixta (OLTP y DSS) se identifican los factores I/O Cost, Cost, Temp Space, Bytes y CPU Cost (véase la figura 6).

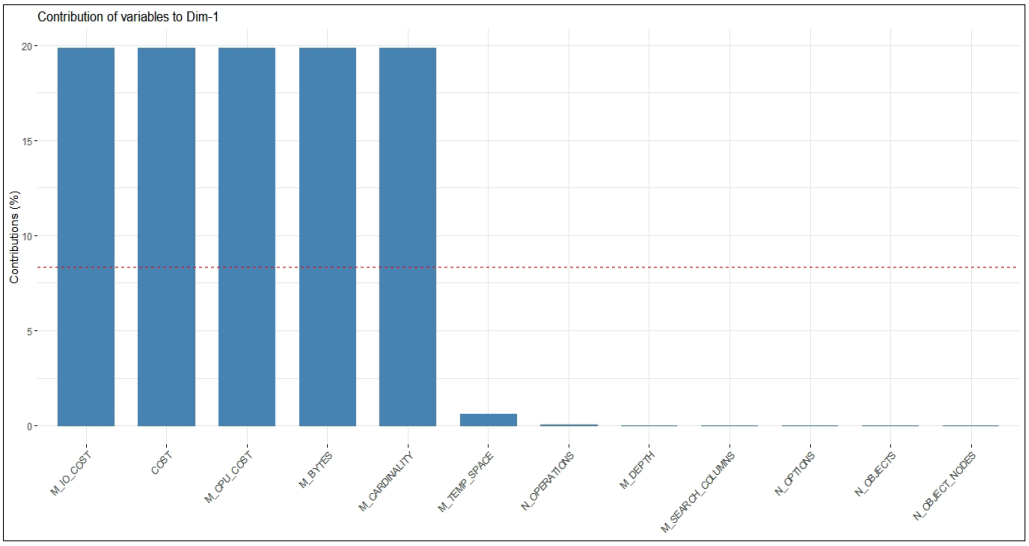

Figura 4. Variables representativas PCA1 de planes de ejecución para base de datos DSS
Elaboración propia

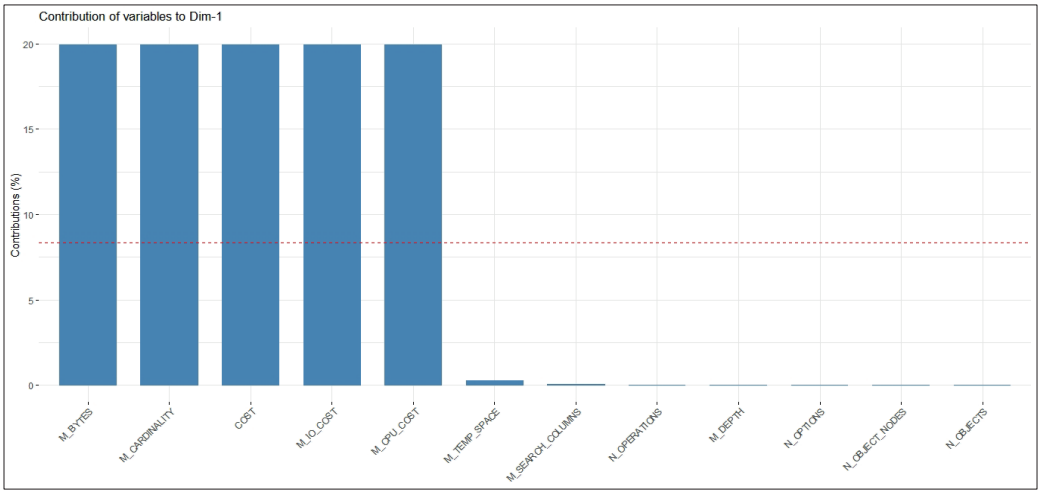

Figura 5. Variables representativas PCA1 de planes de ejecución para base de datos OLTP
Elaboración propia

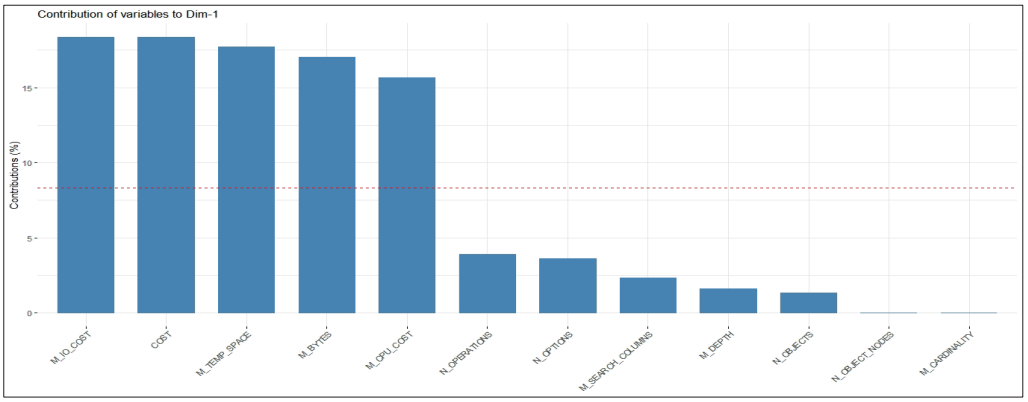

Figura 6. Variables representativas PCA1 de planes de ejecución para base de datos con carga de trabajo mixta OLTP y DSS
Elaboración propia

Las variables que se repiten en todas las cargas de trabajo presentadas, y que a su vez representan los factores que influyen en el rendimiento, son: Cost, CPU Cost, I/O Cost y Bytes. Cabe mencionar que el factor Cost incluye CPU Cost y I/O Cost. Se puede considerar a Cost como la suma de CPU Cost y I/O Cost. Bytes es el volumen de datos procesados. Un modelo de regresión lineal básico consideraría los factores CPU Cost, I/O Cost y Bytes como variables dependientes. Un modelo de regresión lineal más simplificado consideraría los factores Cost y Bytes como variables dependientes.

4.2 Aplicación de Regresión de Componentes Principales (PCR)

Esta sección se trabajó con la información recolectada de la carga de trabajo perteneciente a la base de datos de tipo DSS. Esta información se dividió en dos conjuntos: entrenamiento (*training*) y evaluación (*testing*). Los componentes principales del conjunto de entrenamiento se ilustran en la figura 7.

Importance of components:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11
Standard deviation	2.2426	1.9764	0.9905	0.68879	0.54743	0.48280	0.27397	0.0339	0.002592	0.0006782	0.0003964
Proportion of Variance	0.4572	0.3551	0.0892	0.04313	0.02724	0.02119	0.00682	0.0001	0.000000	0.0000000	0.0000000
Cumulative Proportion	0.4572	0.8123	0.9015	0.94464	0.97188	0.99307	0.99989	1.0000	1.000000	1.0000000	1.0000000

Standard deviations (1, .., p=11):

[1] 2.2426248476 1.9763698289 0.9905427760 0.6887873756 0.5474293321 0. 4828008273 0.2739727423 0.0338989795 0.0025920034 0.0006781921 0.0003964415

Rotation (n x k) = (11 x 11):

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11
N_OBJECTS	0.01180868	-0.45837140	-0.04735051	0.027010625	0.157028575	0.8398943294	-0.238121859	0.0014349197	-0.0001397019	-1.022875E-04	-2.230106E-05
N_OPERATIONS	0.03115959	-0.45807218	0.06614158	0.163116812	0.533361073	-0.4889867369	-0.484376197	-0.0038435533	-0.0001011616	-6.942278E-05	1.284814E-06
N_OPTIONS	0.01194916	-0.48652532	0.02098440	0.094954195	0.235955791	-0.0757833776	0.832030299	0.0009361658	0.0002847114	2.208943E-04	5.087280E-05
M_DEPTH	0.01641463	-0.42837801	-0.02730677	0.482665291	-0.741945458	-0.1429070925	-0.107730282	0.0002793123	0.0000498423	-5.257574E-05	-1.300484E-05
M_SEARCH_COLUMNS	-0.01301118	-0.39857307	-0.05658218	-0.853126447	-0.282352069	-0.1597372544	-0.068562353	-0.0021088065	-0.0001065843	-2.389062E-05	-1.321920E-05
M_CARDINALITY	0.44523895	0.01395551	-0.03630805	-0.008464530	-0.004304858	0.0036340894	0.006242547	-0.8800407576	-0.1599347520	7.364380E-03	1.339231E-03
M_BYTES	0.44546644	0.01263093	-0.03621469	-0.009898311	-0.003212137	0.0005678458	0.003627759	0.0649137280	0.8910088240	-4.244321E-02	-7.408337E-03
M_CPU_COST	0.44544886	0.01239201	-0.03657886	-0.010098423	-0.002911585	-0.0001603897	0.003279041	0.2713850931	-0.2055420442	7.536631E-01	3.406714E-01
M_IO_COST	0.44545164	0.01238880	-0.03640829	-0.010090366	-0.002856980	-0.0002236475	0.003555078	0.2716071767	-0.2541752483	-6.608515E-02	-8.106962E-01
M_TEMP_SPACE	0.07969527	-0.01324418	0.99114905	-0.048796338	-0.070238483	0.0614351600	-0.002803362	0.0003042342	-0.0002103413	1.262246E-04	5.841310E-05
COST	0.44545130	0.01239668	-0.03641268	-0.010100189	-0.002858194	-0.0002345683	0.003642974	0.2717937079	-0.2714247463	-6.525097E-01	4.760810E-01

Figura 7. Componentes principales del conjunto de entrenamiento
Elaboración propia

Los tres primeros componentes principales concentran el 90,15 % de la varianza, y los ocho primeros componentes principales representan el 100 % de la varianza, tal como se observa en la figura 8.

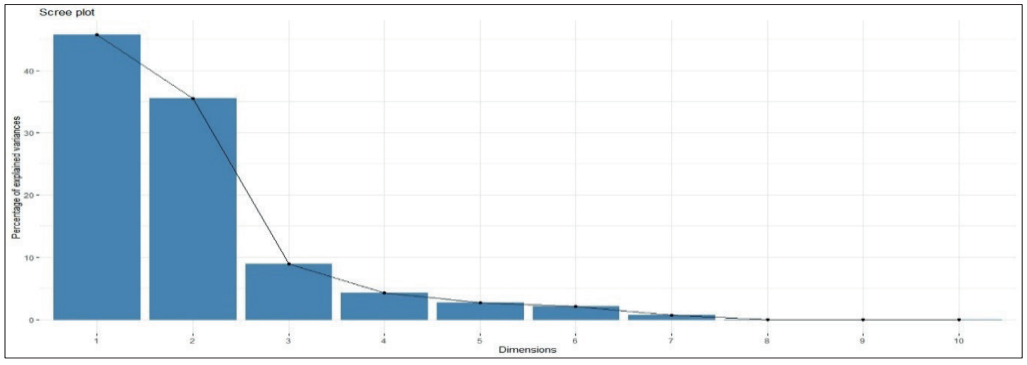

Figura 8. Varianza por cada componente principal
Elaboración propia

Se ejecutó una regresión lineal con los componentes principales del conjunto de entrenamiento, donde los resultados de tiempo de respuesta también fueron normalizados. Las variables independientes son los componentes principales y la variable dependiente el tiempo de respuesta (*elapsed time*). En la figura 9 se aprecian los resultados.

```
Call:
lm(formula = train.y.norm ~ ., data = ols.data)

Residuals:
     Min       1Q   Median       3Q      Max
-4.4513 -0.1070 -0.0555  0.0169  7.5178

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.206e-15  2.281e-02    0.000 1.000000
PC1          1.018e-02  1.018e-02    1.000 0.317512
PC2         -3.206e-02  1.155e-02   -2.776 0.005676 **
PC3          1.015e-01  2.304e-02    4.403 1.27e-05 ***
PC4          1.208e-01  3.314e-02    3.645 0.000292 ***
PC5         -1.869e-01  4.170e-02   -4.483 8.86e-06 ***
PC6          2.188e-02  4.728e-02    0.463 0.643748
PC7          1.253e-01  8.332e-02    1.504 0.133055
PC8          2.175e-01  6.734e-01    0.323 0.746857
PC9         -6.395e+01  8.806e+00   -7.262 1.22e-12 ***
PC10        -1.142e+03  3.366e+01  -33.939  < 2e-16 ***
PC11         4.555e+02  5.758e+01    7.911 1.28e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5587 on 588 degrees of freedom
Multiple R-squared:  0.6936,    Adjusted R-squared:  0.6879
F-statistic:   121 on 11 and 588 DF,  p-value: < 2.2e-16
```

Figura 9. Regresión lineal con componentes principales del conjunto de entrenamiento
Elaboración propia

En la regresión lineal se han utilizado los once componentes principales. Seguidamente, se usa el algoritmo PCR implementado en lenguaje R empleando los datos normalizados de entrenamiento.

```
Data:   X dimension: 600 11
        Y dimension: 600 1
Fit method: svdpc
Number of components considered: 11
TRAINING: % variance explained
             1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps  10 comps  11 comps
X            45.72151  81.2309   90.151   94.464   97.188   99.307   99.989  100.000  100.000     100.0    100.00
train.y.norm   0.05215   0.4537    1.464    2.156    3.203    3.214    3.332    3.338    6.085      66.1     69.36
```

Figura 10. Regresión lineal de los once componentes principales
Elaboración propia

Se observa que utilizando los ocho componentes principales se logra explicar el 100 % de la varianza. Asimismo, considerando los datos de entrenamiento, la predicción del modelo es similar a los datos registrados, donde gran parte de los datos reales y proyectados por el algoritmo PCR se encuentran en la zona inferior izquierda de la figura 11. Nótese la concentración de puntos alrededor de (0,0) al ser datos normalizados.

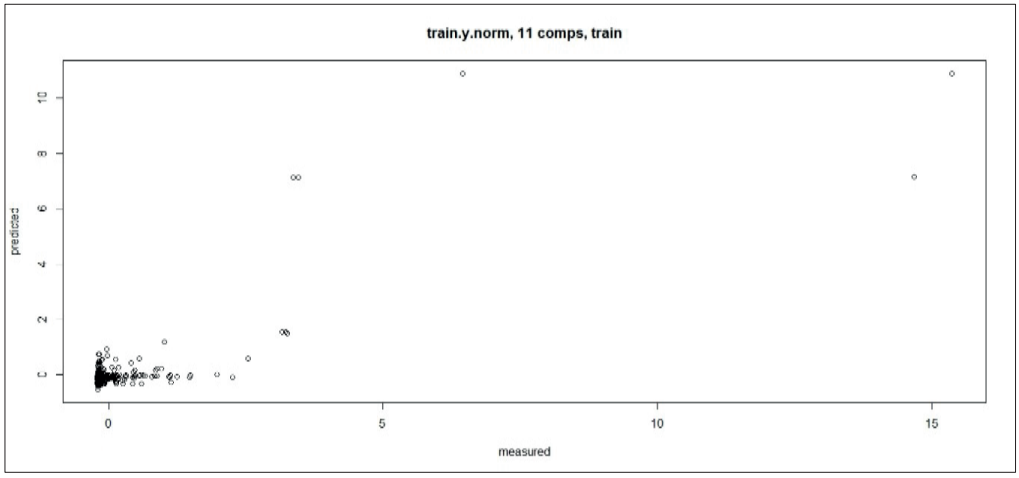

Figura 11. Comparación de las predicciones del tiempo de respuesta sobre los datos de *training*
Elaboración propia

Con los resultados de la regresión obtenida se predicen los valores de tiempo de respuesta para el conjunto de datos de evaluación *(testing*) de dos maneras: utilizando la función *Predict* de lenguaje R sobre el modelo obtenido por PCR aplicado sobre los datos de entrenamiento (y.pred.test1), y aplicando los coeficientes BETA obtenidos por la regresión lineal de los componentes principales (y.pred.test2). Se observa en la figura 12 que ambas predicciones son similares.

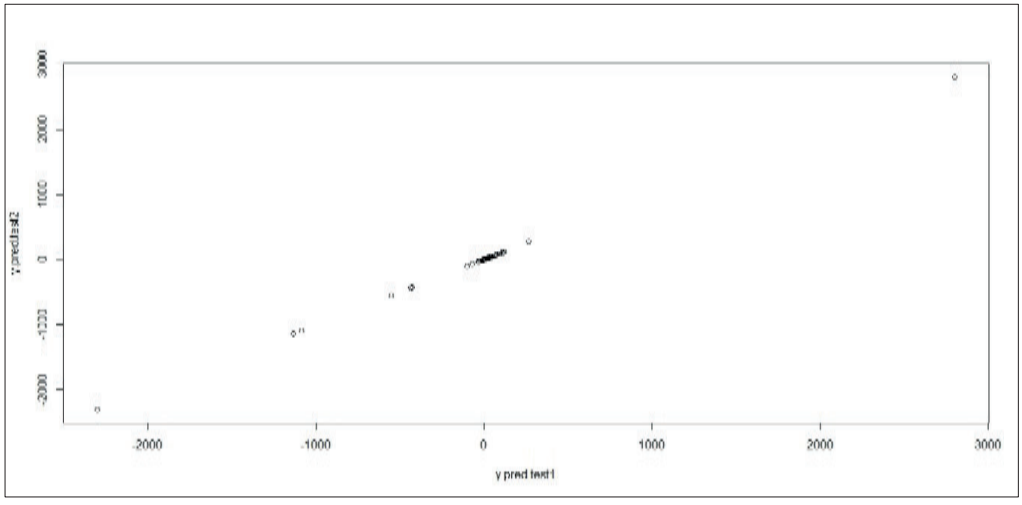

Figura 12. Comparación de las predicciones del tiempo de respuesta sobre los datos de *testing*
Elaboración propia

Considerando los datos de evaluación, las predicciones del modelo son similares a los datos registrados, donde gran parte de los datos reales y proyectados por el algoritmo PCR se encuentran en la zona central izquierda de la figura 13. Nótese la concentración de puntos alrededor de (0,0) al ser datos normalizados.

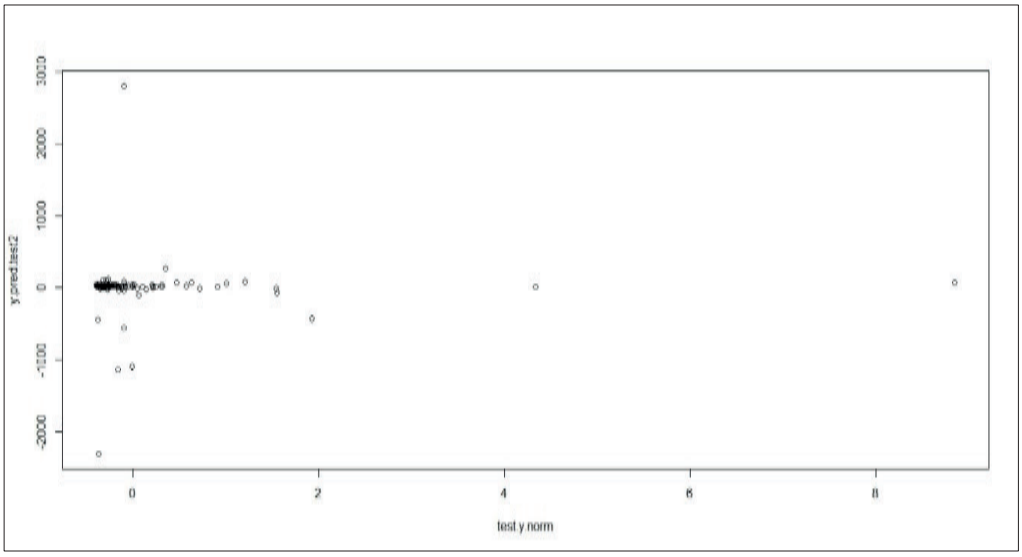

Figura 13. Comparación de las predicciones del tiempo de respuesta sobre los datos de *testing*
Elaboración propia

El error cuadrático medio (MSE) de las predicciones comparadas con los datos reales de *testing* de hasta ocho componentes principales es menor a 1, y a partir de nueve componentes principales el valor de MSE se eleva considerablemente (véase la tabla 2).

Tabla 2
MSE (error cuadrático medio) por número de componentes principales

Número de componentes	MSE
1	0,992594
2	0,987339
3	0,975613
4	0,962656
5	0,982049
6	0,982324
7	0,983122
8	0,975880

(*continúa*)

(*continuación*)

9	256,536200
10	207 763,000000
11	137 537,900000

Elaboración propia

Las predicciones versus los datos reales de *testing* considerando ocho componentes principales se visualizan en la figura 14. Se observa cercanía de valores, a excepción de algunos pocos *outliers*, comunes en modelos de regresión lineal.

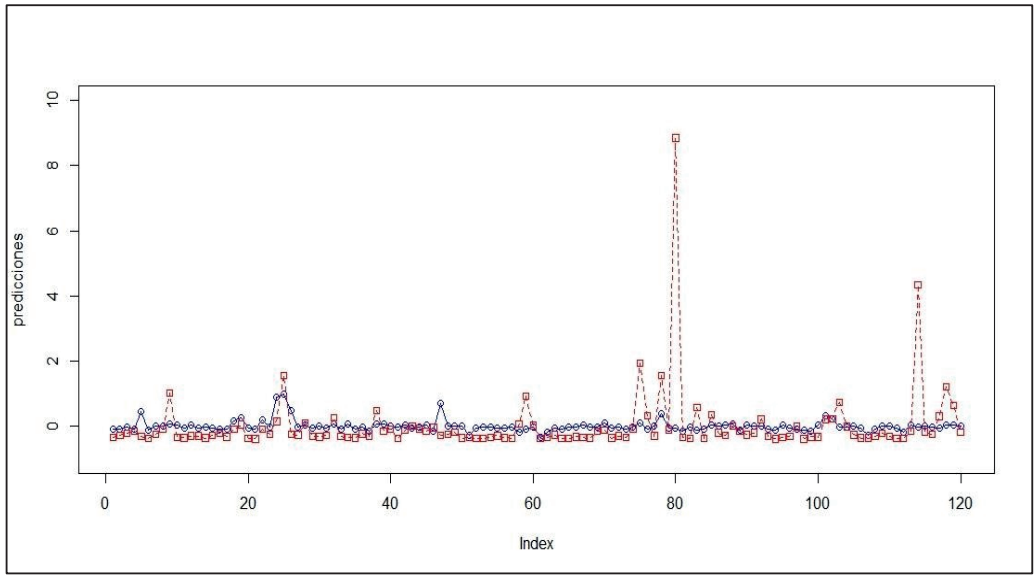

Figura 14. Valores reales (color azul) versus proyectados (color rojo) para el conjunto de datos de *testing*
Elaboración propia

5. CONCLUSIONES

El presente estudio permite determinar los factores que dominan el rendimiento de un sistema de gestión de base de datos relacional visto como caja negra, sin necesidad de entrar en los detalles internos de su arquitectura. Para lograr el objetivo propuesto se utilizaron algoritmos de aprendizaje de máquina, específicamente el Análisis de Componentes Principales (PCA) .Los resultados arrojan como factores influyentes a Cost I/O Cost y CPU Cost y Bytes (volumen de datos procesados).

La metodología presentada, previa adaptación de las estructuras de datos de entrada, puede ser portada a otras plataformas y motores de bases de datos relacionales. Los factores tales como I/O Cost, CPU Cost, Bytes y Cost son conceptos transversales a todas las versiones

de bases de datos SQL y permiten abstraer complejidades propias de cada plataforma, tales como arquitectura de procesamiento y memoria, subsistemas de discos, y redes.

El algoritmo *Principal Components Regression* (PCR) podría considerarse como un método para estimar las variables de respuesta de un sistema, aun considerando un gran número de variables independientes. Puesto que el algoritmo PCA ayuda a descartar variables redundantes, si los componentes principales se emplean como variables independientes del modelo, el algoritmo PCR puede mejorar la fidelidad de sus resultados.

Una estimación inicial de rendimiento consideraría los factores CPU Cost, I/O Cost y Bytes como variables dependientes para la estimación de parámetros de rendimiento. Una estimación más simplificada consideraría los factores Cost y Bytes como variables dependientes.

Para una aplicación válida del algoritmo PCR se deben cumplir los prerrequisitos de la regresión por mínimos cuadrados, tales como la condición de no colinealidad. Es importante que la interpretación de los resultados iniciales, intermedios y finales sea apoyada por especialistas en estadística y tecnología de la información.

La efectividad de la metodología se apoya en contar con estimaciones de costo valederas, y estas se obtienen cuando las estadísticas de los objetos de un motor de base de datos están actualizadas, o en su defecto representan estadísticamente al conjunto de datos que se maneja.

REFERENCIAS

Anderson, M. R., y Cafarella, M. (2016). Input Selection for Fast Feature Engineering. *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, 577-588. https://doi.org/10.1109/ICDE.2016.7498272

Badrinath Krishna, V., Weaver, G. A., y Sanders, W. H. (2015). PCA-Based Method for Detecting Integrity Attacks on Advanced Metering Infrastructure. En J. Campos y B. Haverkort (Eds.), *Quantitative Evaluation of Systems: 12th International Conference, QEST 2015* (pp. 70-85). Springer, Cham. https://doi.org/10.1007/978-3-319-22264-6_5

Bontempi, G., y Kruijtzer, W. (2002). A Data Analysis Method for Software Performance Prediction. *Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition*, 971-976. https://doi.org/10.1109/DATE.2002.998417

Brauckhoff, D., Salamatian, K., y May, M. (2009). Applying PCA for Traffic Anomaly Detection: Problems and Solutions. *IEEE INFOCOM 2009*, 2866-2870. https://doi.org/10.1109/infcom.2009.5062248

De, P., Sinha, A. P., y Vessey, I. (2001). An Empirical Investigation of Factors Influencing Object-Oriented Database Querying. *Information Technology and Managemen*t, *2*, 71-93. https://doi.org/10.1023/A:1009934820999

Duggan, J., Cetintemel, U., Papaemmanouil, O., y Upfal, E. (Junio del 2011). Performance Prediction for Concurrent Database Workloads. *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (SIGMOD '11)*, 337-348. https://doi.org/10.1145/1989323.1989359

Fortier, P. J., y Michel, H. E. (2003). *Computer Systems Performance Evaluation and Prediction*, Digital Press.

Ganapathi, A., Kuno, H., Dayal, U., Wiener, J. L., Fox, A., Jordan, M., y Patterson, D. (Marzo del 2009). Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning. *2009 IEEE 25th International Conference on Data Engineering*, 592-603. https://doi.org/10.1109/icde.2009.130

Ganapathi, A. S., Kuno, H. A., y Dayal, U. (2015). *Predicting Performance of Multiple Queries Executing in a Database* (US 9,189,523 B2). United States Patent and Trademark Office.

Giusto, P., Martin, G., y Harcourt, E. (2001). Reliable Estimation of Execution Time of Embedded Software. *Proceedings Design, Automation and Test in Europe. Conference and Exhibition 2001*, 580-588. https://doi.org/10.1109/DATE.2001.915082

Hadi, A. S., y Ling, R. F. (1998). Some Cautionary Notes on the Use of Principal Components Regression. *The American Statistician*, *52*(1), 15-19. https://doi.org/10.1080/00031305.1998.10480530

James, G., Witten, D., Hastie, T., y Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer.

Kleinrock, L. (1976). *Queuing Systems, Volume II: Computer Applications*. John Wiley & Sons.

Lam, H. T., Thiebaut, J. M., Sinn, M., Chen, B., Mai, T., y Alkan, O. (2017). *One Button Machine for Automating Feature Engineering in Relational Databases.* arXiv. http://arxiv.org/abs/1706.00327

Lee, H., Park, Y. M., y Lee, S. (2015). Principal Component Regression by Principal Component Selection. *Communications for Statistical Applications and Methods*, *22*(2), 173-180. https://doi.org/10.5351/CSAM.2015.22.2.173

Mikolajczyk, Y., y Schmid, C. (2005). A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(10), 1615-1630.

Panda, R., Erb, C., Lebeane, M., Ryoo, J. H., y John, L. K. (Octubre del 2015). Performance Characterization of Modern Databases on Out-of-Order CPUs. *2015 27th International Symposium on Computer Architecture and High Performance Computing*, 114-121. https://doi.org/10.1109/SBAC-PAD.2015.31

Schkolnick, M., y Tiberio, P. (1985). Estimating the Cost of Updates in a Relational Database. *ACM Transactions on Database Systems (TODS)*, *10*(2), 163-179. https://doi.org/10.1145/3857.3863

Shawe-Taylor, J. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press. https://doi.org/10.1017/CBO9780511809682

Shyu, M. L., Chen, S. C., Sarinnapakorn, K., y Chang, L. (2003). A Novel Anomaly Detection Scheme Based on Principal Component Classifier. *Defense Technical Information Center*. https://apps.dtic.mil/sti/citations/ADA465712

Smith, C. U., y Williams, L. G. (2000). Performance and Scalability of Distributed Software Architectures. *An SPE approach. Parallel and Distributed Computing Practices*, *3*(4).

Woodside, M., Franks, G., y Petriu, D. C. (Mayo del 2007). *The future of Software Performance Engineering. Future of Software Engineering (FOSE'07)*, 171-187. https://doi.org/10.1109/FOSE.2007.32

Yu, P. S., Chen, M. S., Heiss, H. U., y Lee, S. (1992). On Workload Characterization of Relational Database Environments. *IEEE Transactions on Software Engineering, 18*(4), 347-355. https://doi.org/10.1109/32.129222