IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Complementos de Bases de Dados

2020/2021

Licenciatura em Engª. Informática

Relatório Técnico

Turma: SW-03

Horário de Laboratório: 5ª feira – 17h:30min

Docente: Luís Cassaca

Grupo

Nº190221032, Tiago Branco

1. Sumário Executivo

Com este projeto pretende-se modelar uma base de dados que providencie um serviço completo de gestão aos responsáveis por uma rede de escolas, permitindo a diferenciação de utilizadores e respetivas autorizações de acesso ao sistema, diferenciação entre dados de histórico estáticos e dados sujeitos a alterações, integridade de dados sensíveis através de uma política de backups eficiente e segura, suporte a acessos e transações concorrentes que mantenham a integridade dos dados e a performance do sistema através de uma boa política de isolamento transacional, estes entre outros de forma a construir um serviço de gestão completo.

Para tal será apresentada uma estruturação de um novo modelo relacional em SQL Server que concretizará vários dos aspetos falados, assim como um planeamento teórico para a gestão e manutenção do mesmo.

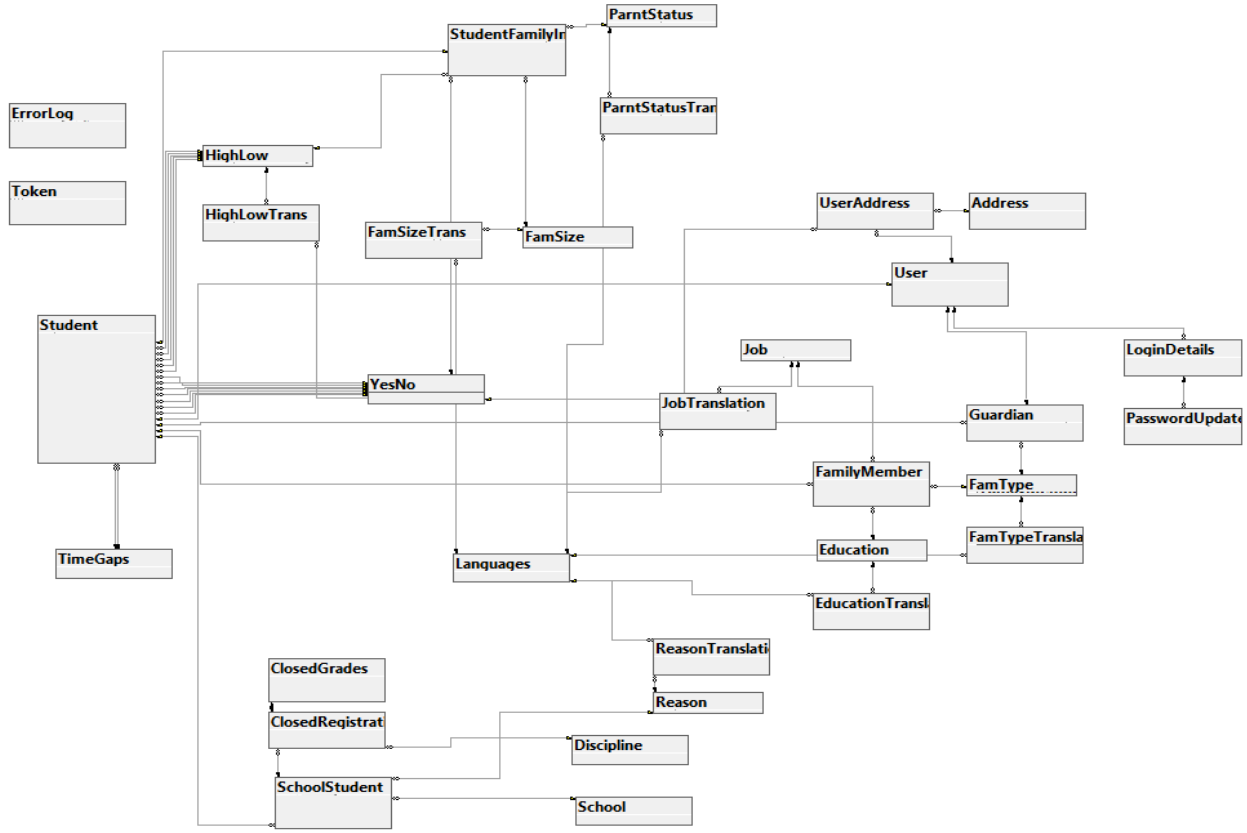
A demonstração em SQL Server encontra-se separada em duas fases de modo a facilitar a compreensão da mesma.

2. Especificação de Requisitos

2.1 Requisitos funcionais

ID	Descrição	Implementado (S/N)
RF01	*O sistema deverá permitir a estudantes e encarregados de educação verificar as notas dos estudantes da sua escola.*	*S*
RF02	Manter um log de erros	S
RF03	Inserir automaticamente alunos repetentes na abertura do novo ano	S
RF04	No fecho do ano, inserir dados em tabelas de histórico	S

3. Modelo Relacional (*Modelo de dados*)



4. Dimensionamento do Layout

Projeções feitas tendo em consideração a modelação da base de dados 'Fase2' e as necessidades de ambas as fases, incluindo tamanho dos índices non-clustered.

A projeção de necessidades quanto à tabela de moradas é 1.5*número de estudantes, devido às possibilidades de moradas múltiplas para cada estudante e/ou pais do estudante, apesar disso não se encontrar refletido nas populações dos exemplos em SQL Server.

Nome Tabela	Dimensão do Registo	Nº de Registos (inicial/final)
SchoolStudent	*12 bytes*	*150.000*
Student	24 bytes	80.000
LoginDetails	232 bytes	160.000
User	108	160.000
Guardian	9	80.000
ClosedRegistrations	9	300.000
ClosedGrades	7	300.000
School	31	4
Address	54	120.000
ErrorLog	212	0
Discipline	31	10
Education	1	5
EducationTranslation	42	10
FamilyMember	7	160.000
Famsize	1	2

FamsizeTrans	32	4
FamType	1	3
FamTypeTranslation	52	6
Highlow	1	5
HighlowTrans	17	10
Job	1	5
JobTranslation	52	10
Languages	3	2
Parntstatus	1	2
Parntstatustrans	32	4
Passwordupdateemail	154	0
Reason	1	4
ReasonTranslation	52	8
StudentFamilyInfo	8	80.000
TimeGaps	31	8
Token	116	0

UserAddress	9	160.000
YesNo	4	2

5. Filegroups

Especificação dos Filegroups considerando as necessidades da Fase2, incluindo índices. A taxa de crescimento é estimada de acordo com as necessidades anuais previstas (número de alunos anuais entre 2000 e 2500).

A taxa de crescimento do filegroup "History" foi obtida considerando que cada aluno tem três disciplinas na sua 1ª inscrição.

Exemplo da concretização em SQL Server em Fase2\Filegroups.

Nome Filegroup	Tabelas associadas	Parâmetros
Filegroup Primary	SchoolStudent, Address, UserAddress, StudentFamilyInfo, FamilyMember	Dimensão inicial: 35mb Dimensão máxima: 150mb Taxa crescimento: 1.5mb
Filegroup Users	LoginDetails, User, Guardian, Student	Dimensão inicial: 23mb Dimensão máxima: 150mb Taxa crescimento: 0.5mb
Filegroup ReadOnly	Discipline, School, Education, EducationTranslation, YesNo, Reason, ReasonTrans, Parntstatus, ParntStatusTrans, Languages, Job, JobTranslation, Famsize, FamsizeTrans, FamType, FamTypeTranslation, Highlow, HighlowTrans	Dimensão inicial: 7mb Dimensão máxima: 50mb Taxa crescimento: 0.3mb
Filegroup History	ClosedRegistrations, ClosedGrades	Dimensão inicial: 25mb Dimensão máxima: 150mb Taxa crescimento: 0.5mb
Filegroup UnpredictGrowth	ErrorLog, PasswordUpdateEmail, Token	Dimensão inicial: 1mb Dimensão máxima: 1000mb Taxa crescimento: 50%

6. Views

Nome	Descrição
dbo.view_getUtilizadores	Esta view permite obter a lista de utilizadores
yrStdntGrowth	Growth in number of students for each year
Grd15Rate	Rate of grades above 15 per year
SchoolBestAvg	Mostra a escola com a melhor média em cada ano

7. Functions

Nome	Tipo	Atributos	Descrição
FinalGradeYr	table	@year int	Returns table with final grades in given year and associated data
getFailedStdtsDisc	table	@year int	Returns students and associated data with final grade < 10 in given year
yearIsOpen	Varchar(5)		Returns true if school year is open
FinalGrade	table		Gets all final grades for each student and associated data from history tables
GrdsAbv15Yr	decimal	@yr int	Gets count of grades above 15 in given year

getPrevCount	decimal	@yr int	Gets count of students in previous year
getAvgGrdYr	table		Devolve uma tabela com a melhor média por ano e escola

8. Stored procedures

Nome	Atributos	Descrição
GenerateToken	@email varchar(100)	Gera um token de 5 dígitos para dado email
RecoverPassword	@tok int, @email varchar(100) @newPass varchar(100)	Processo para mudar a password em loginDetails dado o token correto em menos de 1 hora
UpdatePassword	@email varchar(50) @oldPass varchar(50) @newPass1 varchar(50) @newPass2 varchar(50)	Mudar a password em loginDetails dada a passe antiga correta
InsertRepeatStudents	@year varchar(4) @table varchar(30)	Inserts failed students from past year in open registrations table
OpenSchoolYear	@year int	Opens new year if there's no year open already
CloseSchoolYear	@year int	Closes school year if there is one open and transfers data to history
UpdateGrade	@grade varchar(2) @period char @idRegistration varchar(8) @year smallint	Inserts/updates given grade
RegisterStudentSchool	@idStudent int @idSchool tinyint	Associates school with student in open school year
RegisterStudentDiscipline	@idStudentSchool varchar(8)	Associates student with discipline in given year

	@year int @idDiscipline char	
TotalOpenRegistrations		Total de alunos de alunos inscritos em cada uma das disciplinas no ano aberto face ao ano anterior e a respetiva taxa de crescimento.
AverageGrades	@year int	Selects grade average of given year and comparison with last year
CreateUserViews		Creates views for students and grades for each school
CreateUserRoles		Creates the user roles
GrantUserPermissions	@userID int	Grants a user their permissions in the database

9. Triggers

Nome	Tipo	Tabela	Descrição
trPasswordUpdateEmail	After update	PasswordUpdateEmail	Insere mensagem de confirmação na tabela PasswordUpdateEmail

10. Consultas

10.1 Verificação da conformidade dos dados

Consultas para fase 1 em MigrationTest

10.2 Outras consultas

Verficar número correto de alunos em cada ano (Fase 2):
select schoolYear, count(*) from dbo.SchoolStudent group by schoolYear order by schoolYear

11. Backup e Recuperação

Padrão de backups:

Full backup semanal (na altura de menor interação com a base de dados) - log backup hora a hora durante 3 horas, à 4ª hora backup parcial, parcial diferencial depois do 1º parcial diário, de forma a excluir filegroups "read only" e transações já em backup, melhorando a performance e reduzindo a carga do sistema sem comprometer demasiado a segurança dos dados em questão.

O modelo de recuperação poderá ser o "full" de modo a maximizar a recuperação das alterações feitas à base de dados até ao "crash".

12. Transações

Num caso de concorrência transacional entre CloseSchoolYear, UpdateGradee e OpenSchoolYear poderíamos definir o nível de isolamento como repeatable read, já que, caso o ano esteja aberto, executando o CloseSchoolYear e o updateGrade nesse mesmo ano simultaneamente, o updateGrade deverá aceder 1º ao lock da nota em questão e alterá-la antes do ano fechar, enquanto que o OpenSchoolYear lança erro (ano já aberto).

No entanto, se o updateGrade inserir uma nota nova poderá ocorrer phantom read, apesar disso não ser provável caso a execução seja exatamente simultânea.
Caso o ano esteja fechado, apenas o OpenSchoolYear irá executar.

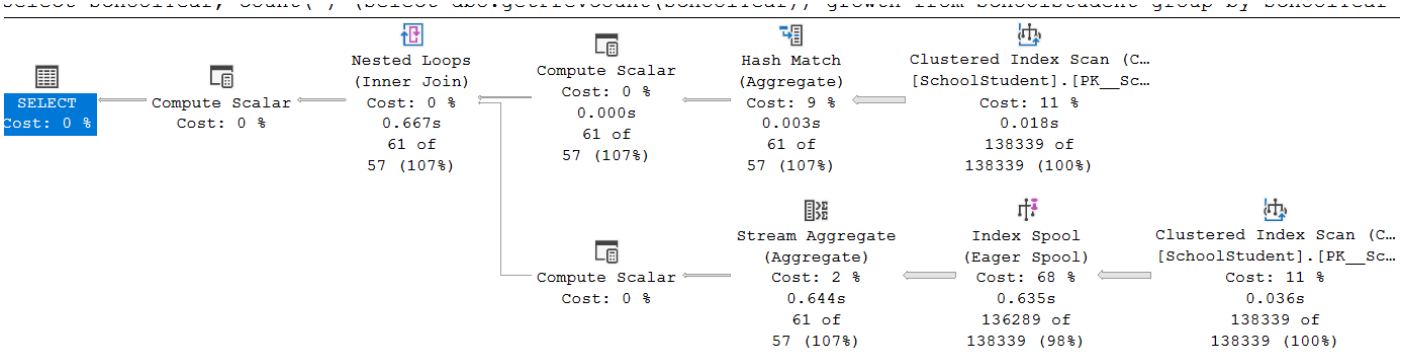
Outro conflito poderia ocorrer entre o TotalOpenRegistrations(Fase1) e a inserção de um novo registo, no qual poderíamos definir o isolamento para read committed de forma que haja uma alta probabilidade do TotalOpenRegistrations ler o novo registo inserido.

13. Índices

Os testes seguintes foram feitos para as consultas dentro das views e não para as views em si. A definição dos índices foi feita com a ajuda do "tuning advisor" do SQL Server e os resultados da execução antes e depois da inserção dos índices é apresentada a seguir:

- Na consulta da view "yrStdntGrowth" a performance é melhorada com o indíce NONCI_year em dbo.SchoolStudent, com um menor número de leituras e menor tempo de execução para apresentar os resultados após inserção dos índices.

Antes:

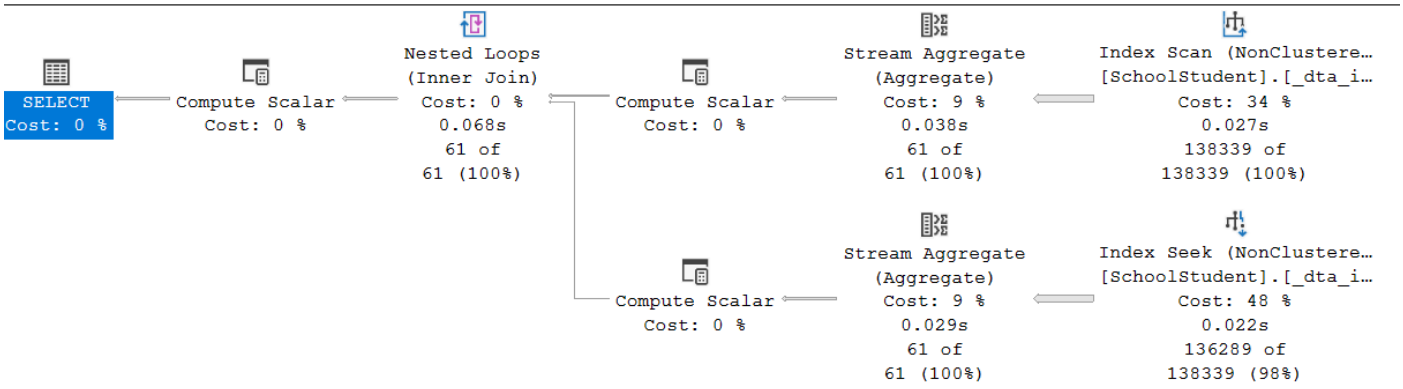



```
(61 rows affected)
Table 'SchoolStudent'. Scan count 2, logical reads 724, physical reads 1,
Table 'Worktable'. Scan count 61, logical reads 278740, physical reads 0,

(1 row affected)

 SQL Server Execution Times:
   CPU time = 672 ms,  elapsed time = 812 ms.
```

Depois:



```
(61 rows affected)
Table 'SchoolStudent'. Scan count 62, logical reads 630,

(1 row affected)

 SQL Server Execution Times:
   CPU time = 78 ms,  elapsed time = 169 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
```

- Na consulta da view GrdsAbv15Yr a performance melhora com os indíces NONCI_grade, NONCI_year, NONCI_Regist e NONCI_SchStd, com a observação que o índice NONCI_Regist provoca o aumento do número de "logical reads" na tabela "ClosedRegistrations", no entanto reduz o número de physical reads e o tempo de execução.

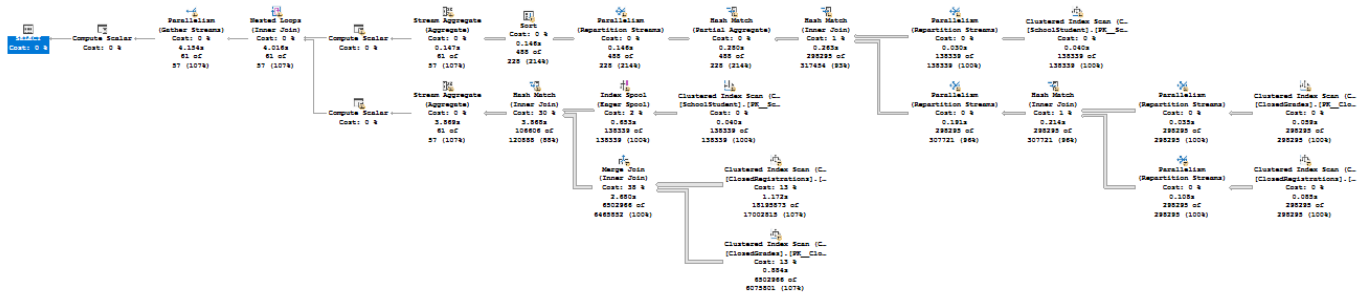
Antes:

```
(61 rows affected)
Table 'SchoolStudent'. Scan count 10, logical reads 1443, physical reads 1, page s
Table 'ClosedGrades'. Scan count 70, logical reads 36723, physical reads 1, page s
Table 'ClosedRegistrations'. Scan count 70, logical reads 41511, physical reads 3,
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, page server rea
Table 'Worktable'. Scan count 61, logical reads 288022, physical reads 0, page ser
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server re

(1 row affected)

 SQL Server Execution Times:
   CPU time = 22715 ms,  elapsed time = 4280 ms.
```

```
Missing Index (Impact 13.1384): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Close
```

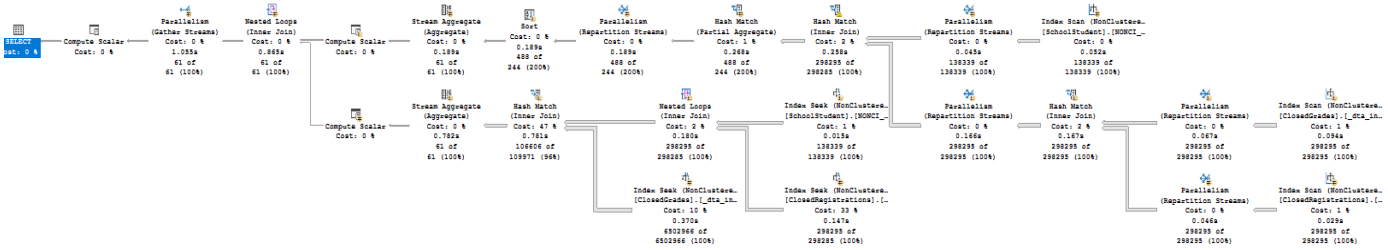


Depois:

```
(61 rows affected)
Table 'SchoolStudent'. Scan count 70, logical reads 1026, physical reads 1, page server
Table 'ClosedGrades'. Scan count 70, logical reads 10197, physical reads 1, page server
Table 'ClosedRegistrations'. Scan count 138348, logical reads 442473, physical reads 3,
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, page server reads 0,
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0

(1 row affected)

 SQL Server Execution Times:
   CPU time = 3049 ms,  elapsed time = 539 ms.
```



- Na view SchoolBestAvg o número de leituras é reduzido com os indices NONCI_grade, NONCI_SchoolYear e NONCI_SchStd, enquanto que a melhoria do tempo de execução é negligenciável.

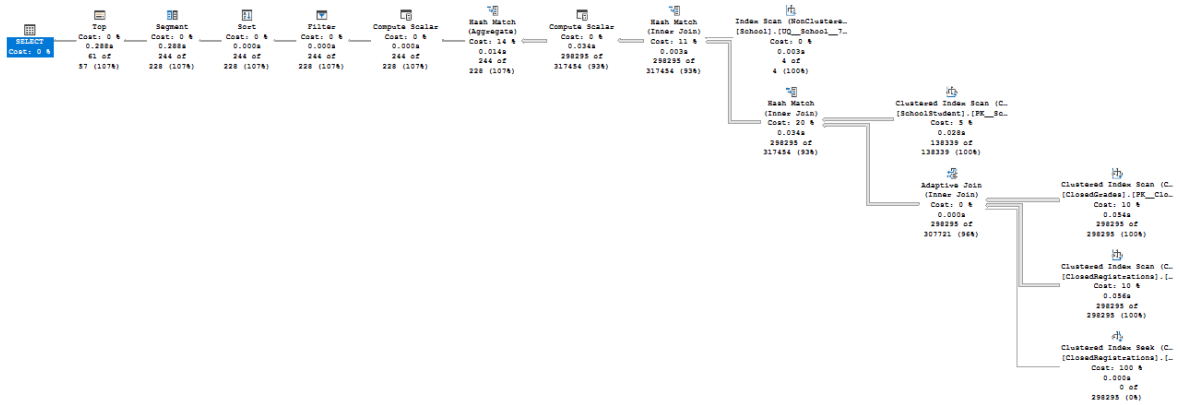
Antes:

```
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 62 ms, elapsed time = 72 ms.

(61 rows affected)
Table 'ClosedRegistrations'. Scan count 1, logical reads 669, physical reads 3,
Table 'ClosedGrades'. Scan count 1, logical reads 592, physical reads 1, page s
Table 'SchoolStudent'. Scan count 1, logical reads 362, physical reads 1, page
Table 'School'. Scan count 1, logical reads 2, physical reads 1, page server re
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page serve
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page serve

(1 row affected)

 SQL Server Execution Times:
   CPU time = 250 ms,  elapsed time = 468 ms.
```

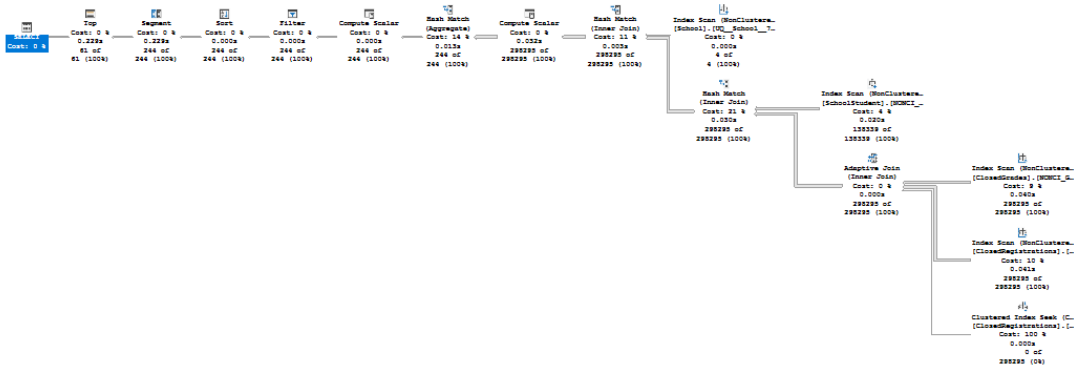


Depois:

```
(61 rows affected)
Table 'ClosedRegistrations'. Scan count 1, logical reads 525, physical reads 0,
Table 'ClosedGrades'. Scan count 1, logical reads 411, physical reads 0, page s
Table 'SchoolStudent'. Scan count 1, logical reads 225, physical reads 0, page
Table 'School'. Scan count 1, logical reads 2, physical reads 0, page server re
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server

(1 row affected)

 SQL Server Execution Times:
   CPU time = 234 ms,  elapsed time = 411 ms.
```



14. Descrição da Demonstração

O aspeto hierárquico da tabela Discipline na base de dados da fase 2 foi removido de modo a simplificar a demonstração.

A procedure "JsonGrades", consulta que gera os dados para extrair para uma coleção em MongoDB, foi feita para mostrar apenas os primeiros 10.000 registos já que não foi possível processar a tabela para um formato json com um número muito elevado de registos.

Ordem de execução Fase1: OldData - Tables – Migrate – MigrateTest – Program - View – ProgramTest

Ordem de execução Fase2: Tables – Program – GenerateData – Index - IndexTest - Views – Permission – Encryption

Extras: MongoDB, Filegroups, Backups

15. Conclusões

Planeamento do modelo tem de ser cuidado pois os erros nesta fase propagam-se no desenvolvimento das funcionalidades programáticas do sistema.

As tabelas em ambas as bases de dados (fases 1 e 2) encontram-se otimizadas de forma a ocupar o menor espaço possível e maximizar a flexibilidade do modelo, o que causa com que muitas das consultas às tabelas tenham que ser feitas através de 'selects' complexos devido ao elevado corelacionamento entre elas.

Índices bem planeados melhoram significativamente a performance das consultas, mas ocupam espaço de memória considerável, pelo que não devem ser sobre utilizados.

No MongoDB os documentos encontram-se num formato simples, o que provoca a repetição de registos com o mesmo estudante. Caso o processo fosse manual ou caso haja uma forma de o fazer de forma automatizada a partir do SQL Server, uma forma mais correta de organizar os dados seria colocando os anos num array e ao nível dos anos um outro array com as disciplinas e por sua vez outro array com as respetivas avaliações, exemplo na página seguinte.

```
db.grades.insert({
	_id:1,
	studentNr:1,
	guardianID:2,
	schoolYear:[{
		year:2016,
		Discipline:[{
			Name: "TIC"
			Grades: [{
				Grade1:8
				Grade2:6
				Grade3:7
				Mean:7
			},
			Name: "Math"
			Grades: [{
				Grade1:10
				Grade2:11
				Grade3:12
				Mean:11
			}]
		},
		year:2017,
		Discipline:[{
			Name: "TIC"
			Grades: [{
				Grade1:12
				Grade2:10
				Grade3:14
				Mean:12
			}]
		}]
	}]
})
```