VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

VEDOUCÍ PRÁCE

doc. Ing. Radim Bača, Ph.D.

STUDENT

Bc. Marie Lašinská
LAS0084

Benchmark ověřující schopnost odstraňovat redundanci z SQL

Semestrální práce

2023/2024

Cíl práce

Vyvinout benchmark, který bude na předpřipravených databázích spouštět SQL dotazy a kontrolovat, zda-li došlo k odstranění nějaké konstrukce.

Motivace

Určit, které databázové systémy jsou efektivnější a optimalizovanější pro zlepšení výkonu aplikací a snížení zátěže na server.

II Databáze

Vybrané databázové systémy:

Microsoft SQL Server, Oracle, MySQL, PostgreSQL

Důvod výběru:

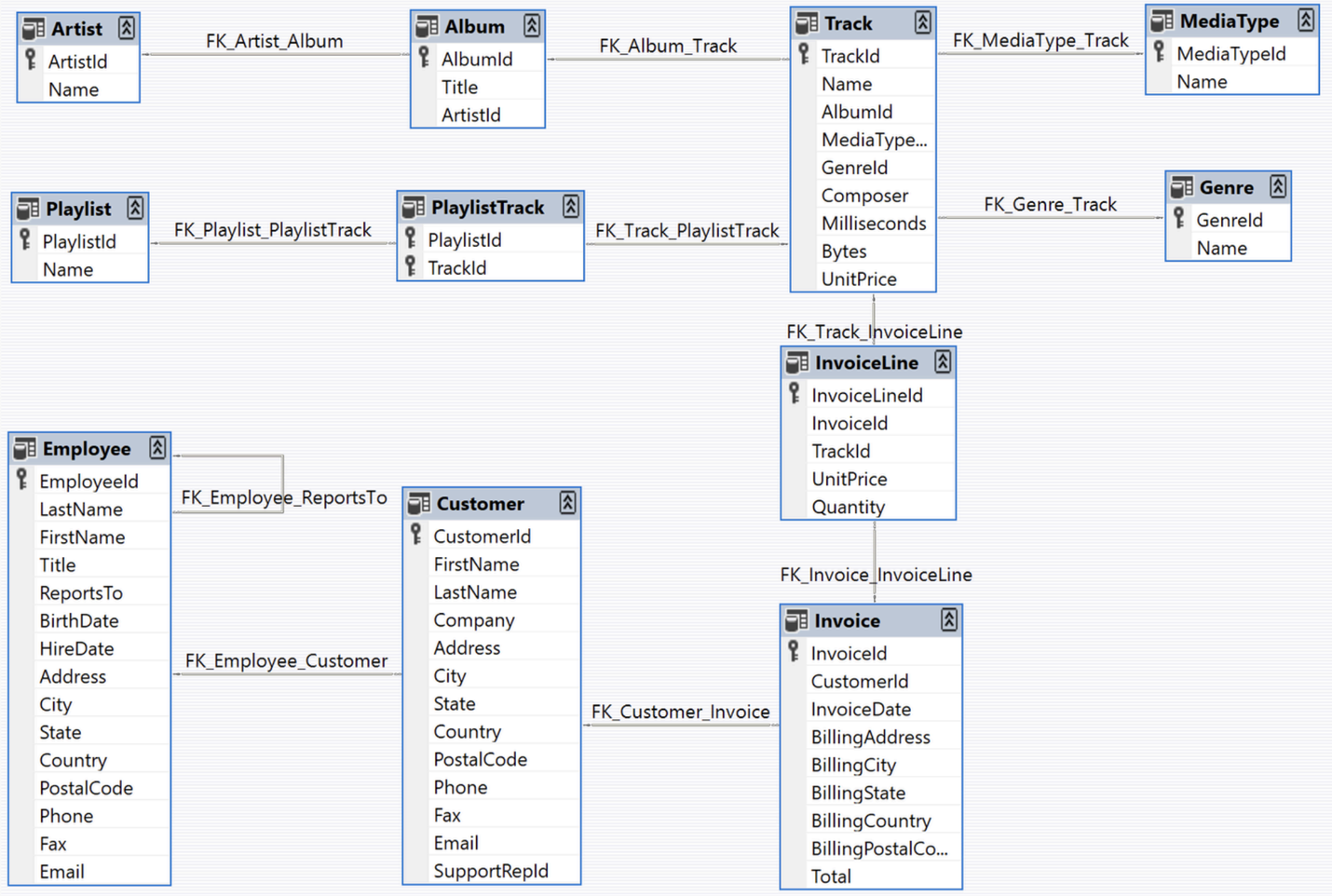
Široké použití v průmyslu a schopnost zobrazit plány vykonání SQL dotazů.

Databáze Chinook

Chinook je volně dostupná ukázková databáze obsahující data s informacemi o skladbách, albech, umělcích atd.

Důvod výběru:

Obsahuje scripty pro všechny vybrané databázové systémy a dostatek dat pro testování různých scénáře dotazů.



Úvod do Problematiky Redundance v SQL

Redundance v SQL dotazech představuje duplikaci informací či operací, které mohou mít negativní vliv na výkon a čitelnost kódu.

Typické příklady zahrnují nadbytečné podmínky ve WHERE klauzulích, redundantní DISTINCT, vnější spojení místo vnitřního nebo také redundantní celé konstrukce jako je spojení či GROUP BY.

Testovacích dotazů bylo vytvořeno 34 a pro přehlednost jsou dotazy rozděleny do 8 kategorií:

Attributes, Distinct, Conditions, Joins and Unions, Aggregations, Grouping, Case, Window.

III Redundantní SQL Dotazy

Příklad 1

Category: Distinct

Source: Semantic errors in SQL queries: A quite complete list.

Reference: 2.2. Error 2

Description: Using distinct on already unique values.

Query with redundancy:

```
SELECT DISTINCT(AlbumId)
FROM Album
```

Query without redundancy:

```
SELECT AlbumId
FROM Album
```

Příklad 2

Category: Joins and Unions

Source: Semantic errors in SQL queries: A quite complete list.

Reference: 2.3. Error 6

Description: Using unnecessary JOIN if we only use attributes from one table.

Query with redundancy:

```
SELECT Invoice.InvoiceId, Invoice.Total
FROM Invoice
JOIN Customer ON Invoice.CustomerId =
Customer.CustomerId
WHERE Invoice.Total < 1
```

Query without redundancy:

```
SELECT Invoice.InvoiceId, Invoice.Total
FROM Invoice
WHERE Invoice.Total < 1
```

Příklad 3

Category: Grouping

Source: Semantic errors in SQL queries: A quite complete list.

Reference: 2.6 Error 19

Description: Using group by on id is unnecessary and redundant because there is always only one tuple per id.

Query with redundancy:

```
SELECT ArtistId, COUNT(*) AS CountArtists
FROM Artist GROUP BY ArtistId
```

Query without redundancy:

```
SELECT ArtistId, '1' AS CountArtists
FROM Artist
```

Implementace v .NET Framework

.NET Framework zvolen pro jeho robustnost při práci s různými databázovými systémy a snadnou integraci s ostatními nástroji jako je např. generování excel souborů.

Funkce benchmarku:

- Podpora všech 4 vybraných databázových systémů
- Načítání SQL dotazů ze souboru
- Možnost výběru přísný vs. volný benchmark
- Extrakce a analýza plánů vykonání SQL dotazů
- Spuštění scriptů s příkazy pro tvorbu a mazání indexů
- Generování textových a Excel souborů s výsledky

Zjednodušený třídní diagram:

Třída Benchmark

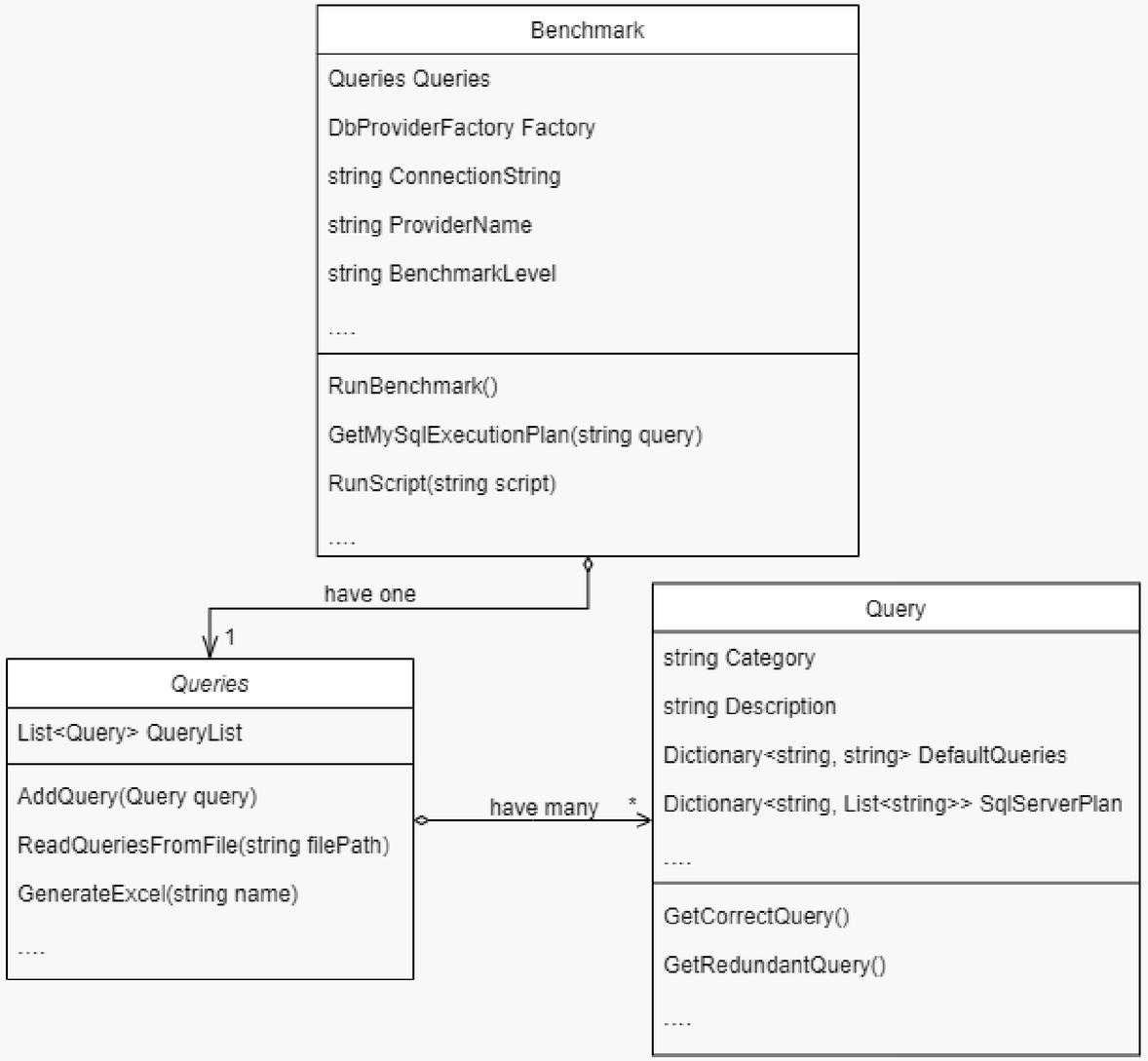
Obsahuje všechny potřebné metody k získání plánů vykonání dotazu na jednotlivých databázových systémech a spuštění scriptů pro vytvoření a smazání indexů.

Třída Queries

Pomocná třída pro načítání a uložení datasetu s dotazy a pro generování excel souborů.

Třída Query

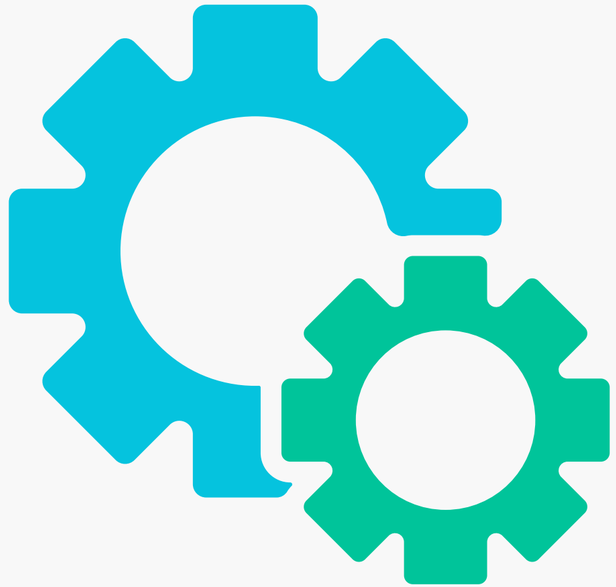
Uchovává veškeré informace o jednotlivých dotazech, včetně vrácených plánů.



Nastavení benchmarku:

Dle úrovně porovnávání:

- **loose** – Porovnává jen provedené operace a objekty na kterých byly provedeny.
- **strict** – Porovnává celé plány vykonání dotazu, včetně statistik jako je počet filtrovaných řádků, IO Cost a velikost výstupu.

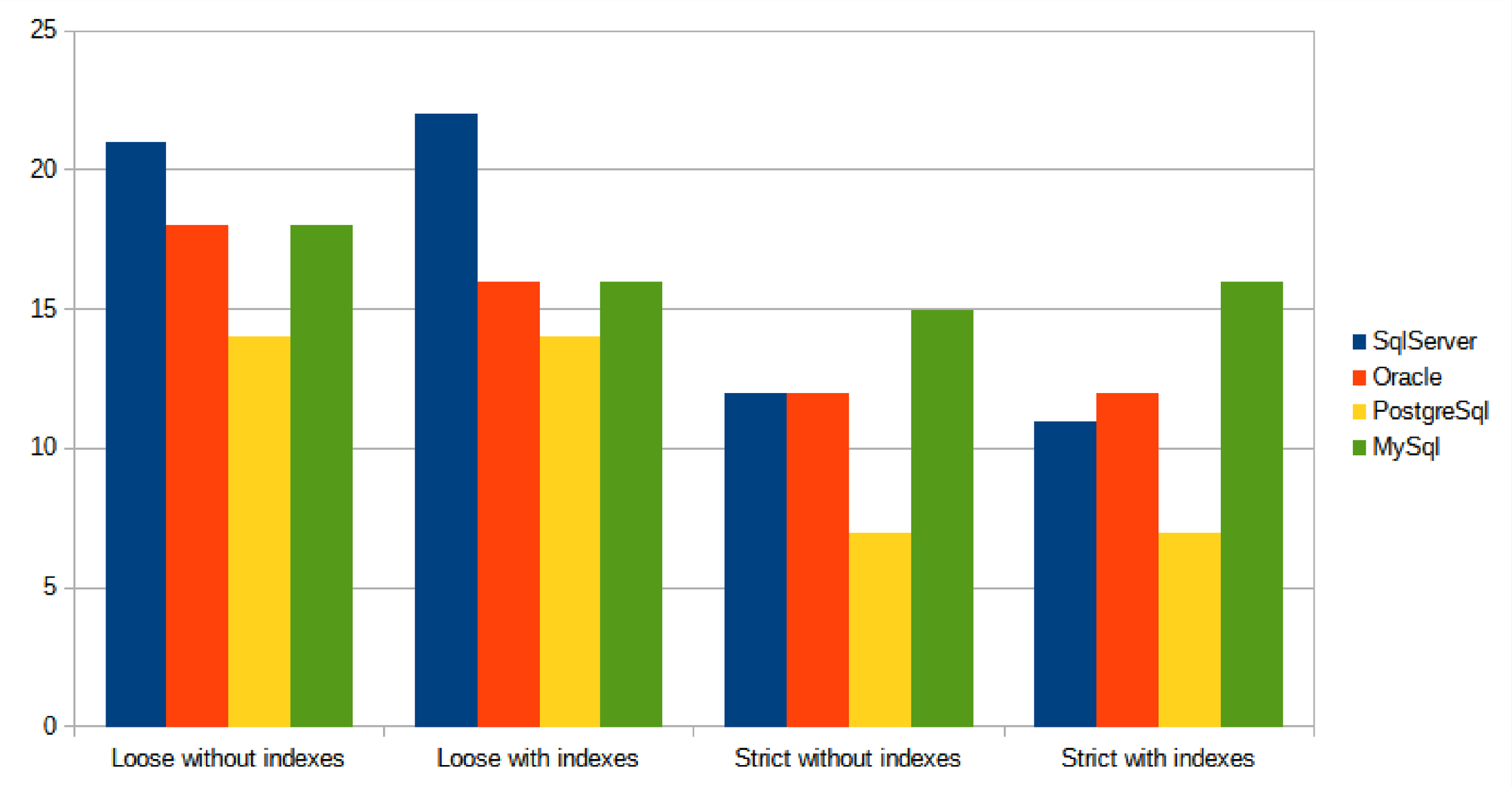


Z pohledu indexů:

- **without indexes** – Bez přidaných indexů na klíčových atributech.
- **with indexes** – S přidanými indexy na klíčových atributech.

V Výsledky

Výsledky celkově (34 dotazů):

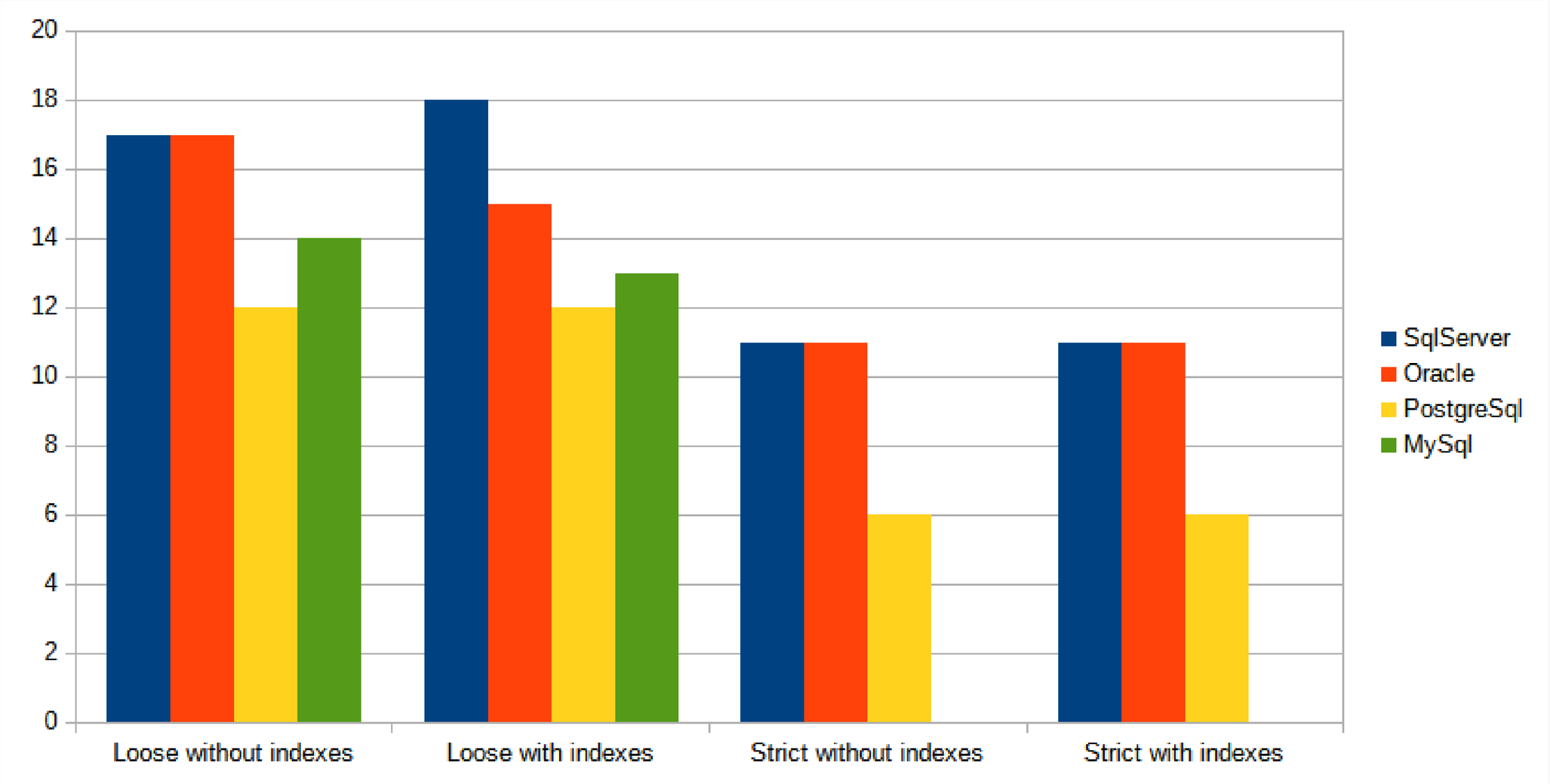


MySQL plán vykonání dotazu

Jelikož MySQL nezobrazuje práci s hashovací tabulkou, přistoupila jsem ke snížení datasetu o 4 dotazy, u kterých právě z tohoto důvodu MySQL vychí lépe než ostatní databázové servery.

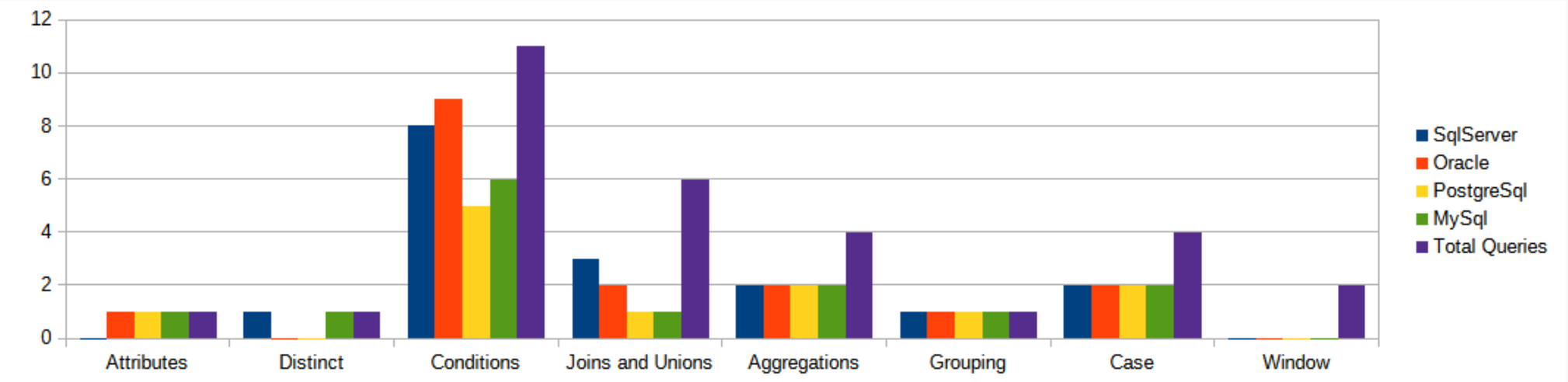
Dále pak MySQL nenabízí tak detailní plán vykonání dotazu, chybí v něm např. cena a velikost dotazu. Proto je z výsledků přísného benchmarku vyloučen.

Výsledky celkově (30 dotazů):

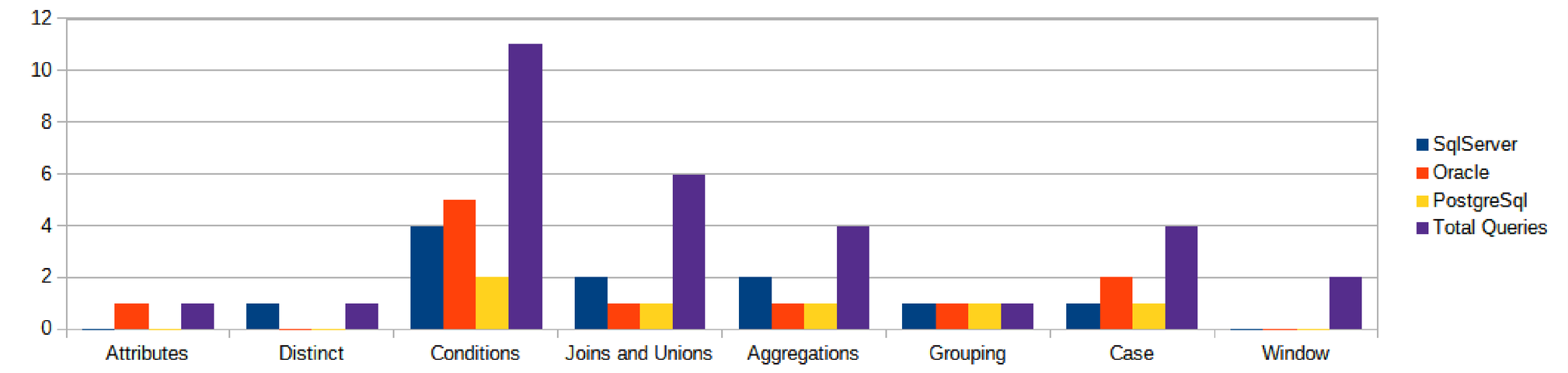


V Výsledky

Po kategoriích bez přidaných indexů (30 dotazů):



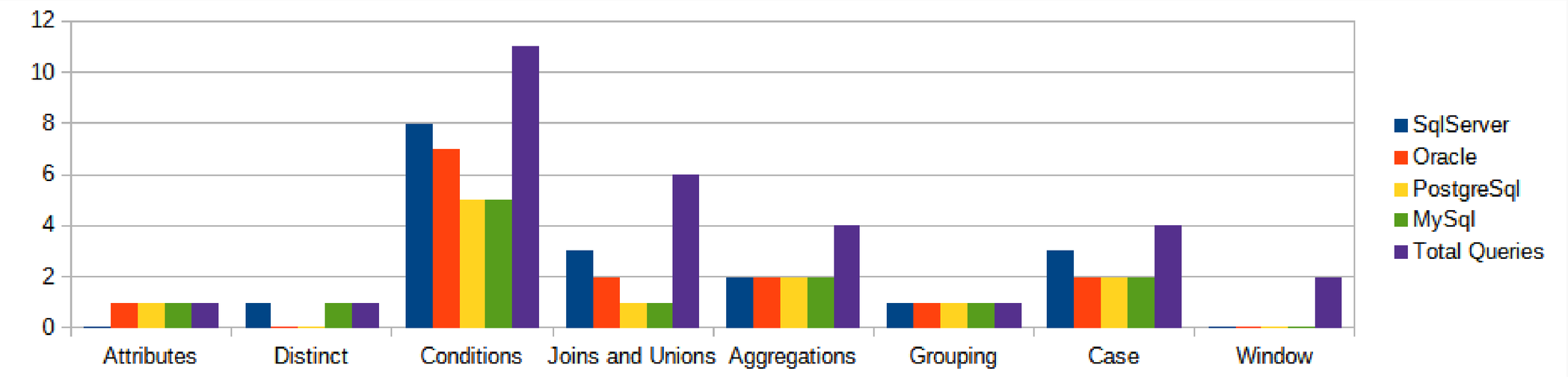
“Loose” benchmark



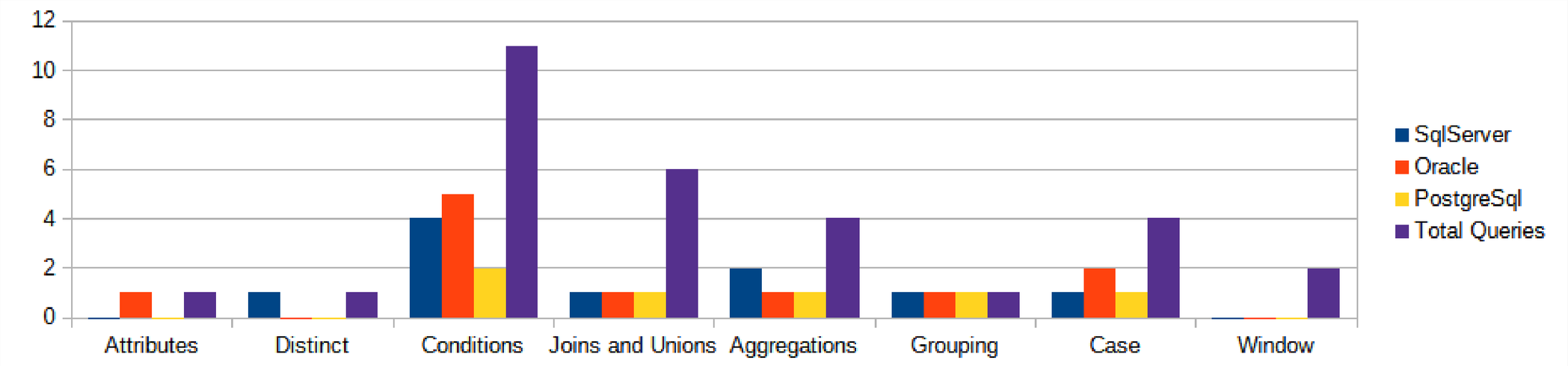
“Strict” benchmark

V Výsledky

Po kategoriích s přidanými indexy (30 dotazů):



“Loose” benchmark



“Strict” benchmark

V Výsledky

Závěry:

SQL Server a Oracle vykazují konzistentní a vysokou schopnost optimalizace dotazů ve všech variantách, což naznačuje robustní optimalizační mechanismy. SQL Server, ale poráží Oracle ve volnějším benchmarku s indexy a má tak obecně nejlepší výsledky a to i v původním datasetu.

PostgreSql má nižší skóre, zejména v přísném benchmarku, což může naznačovat potřebu lepší optimalizace.

MySql se ve volnějším benchmarku drží mezi ostatními databázemi, ale není hodnocen v přísných podmínkách z důvodu nedostatku detailních plánů vykonání dotazu.

VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

VEDOUCÍ

doc. Ing. Radim Bača, Ph.D.

STUDENT

Bc. Marie Lašinská
LAS0084

Děkuji za pozornost.

Dotazy?