Database systems I

Radim Bača

radim.baca@vsb.cz

dbedu.cs.vsb.cz

VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Obsah

- Proč jiné databáze?
- Architektury IS
- Základní orientace v databázových systémech

Vždyť nás učili, že relační jsou ok

- Máme dva typy relačních databází:
  - Open-source - PostgreSQL, MySQL
  - Komerční - SQL Server, Oracle, DB2
- Jedny jsou zdarma a za druhé musíme cálovat
- Tak jaképak copak!

Vždyť nás učili, že relační jsou ok

- Máme dva typy relačních databází:
  - Open-source - PostgreSQL, MySQL
  - Komerční - SQL Server, Oracle, DB2
- Jedny jsou zdarma a za druhé musíme cálovat
- Tak jaképak copak!

Zkusme se na to podívat podrobněji ...

Výhody relačních databázových systémů

- Robustní a elegantní model
  - Referenční integrita (Cizí a primární klíče)
  - Integritní omezení
- Příjemné rozhraní - SQL

Výhody relačních databázových systémů

- Ostatní výhody nalezneme i v jiných databázových systémech
  - Transakční integrita (při řízení souběhu operací)
  - Persistence
  - Řízení přístupu
  - Zotavení po chybě
  - Spolehlivost (dostupnost)

Kdy použít jiné databázové systémy?

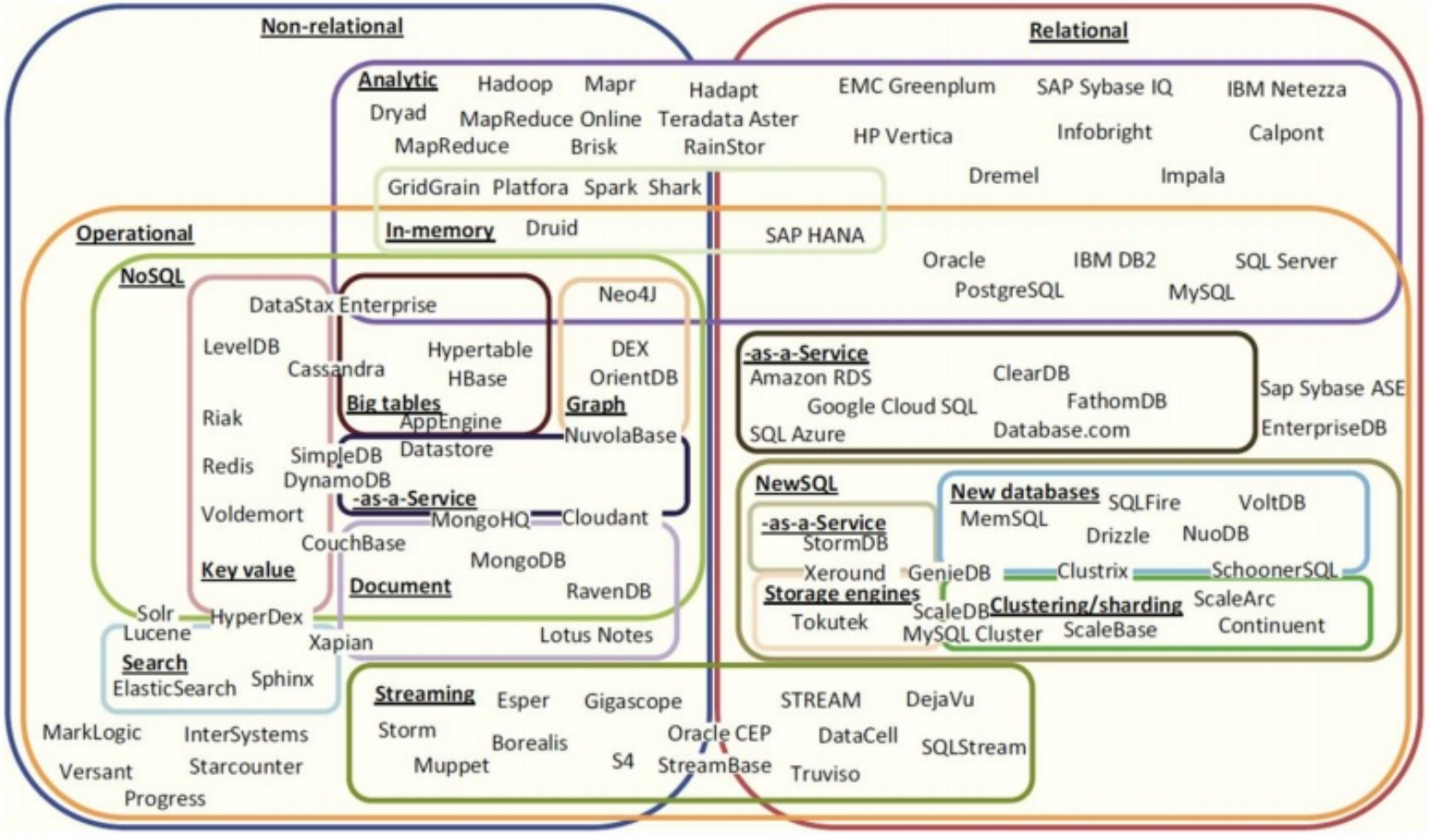
Kdy použít jiné databázové systémy?

Záleží na daném problému

Aspekty výběru databázového systému

- Cílová architektura
- Typ vytížení
- Datový model
- Dostupnost
- Hlavní paměť
- DBaaS

Aspekty výběru databázového systému

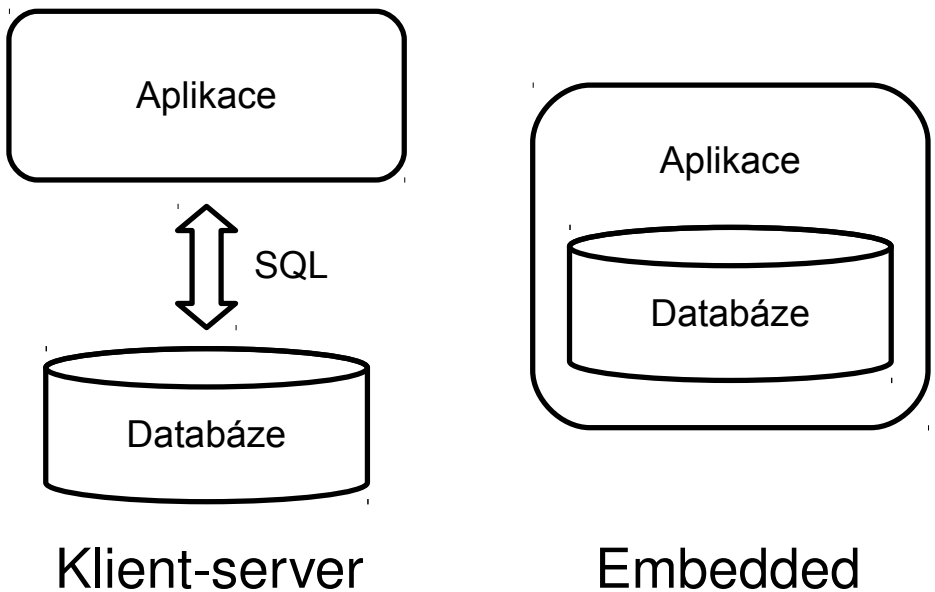


Architektura IS

- S databází uživatelé nepracují přímo přes klienta typu SQL Server Management Studio
- Koncoví uživatelé pracují s IS (aplikací), která komunikaci s databází zprostředkovává a interpretuje
- IS je tedy nějaká aplikace napsaná v jazyce typu Java, .NET
- Rozlišujeme tedy dva typy IS:
  - Web aplikace - aplikace, se kterou pracujeme s využitím nějakého webového prohlížeče
  - Desktopová aplikace - jedná se o aplikaci běžící přímo v OS, většinou se jedná klasickou oknovou aplikaci

Architektura IS

- Dvě základní architektury:
  - Klient-server (Oracle, SQL Server, MySQL, ...)
  - Embedded (SQLite, BerkeleyBD, RocksDB)



Rozdělení dle typické zátěže

- Můžeme rozdělit databázové systémy podle typu operací, který se pro daný systém hodí:
  - Transakční databáze (OLTP) - provádíme opakovaně malé množství podobných dotazů, kde každý dotaz zpracovává poměrně málo záznamů (v porovnání s množstvím dat)
  - Datové sklady (Analytické, OLAP databáze) - provádíme rozsáhlé dotazy, jenž počítají souhrny nad všemi daty
  - Kombinované vytížení (HTAP)

Sloupcové databáze

- Databáze pro datové sklady bývají často odlišně koncipovány (sloupcově orientované databáze)
- Vedle sebe jsou (v paměti i na disku) uloženy hodnoty ze stejného sloupce
- Vertica, MonetDB, Netezza, ...

Sloupcové databáze

- Databáze pro datové sklady bývají často odlišně koncipovány (sloupcově orientované databáze)
- Vedle sebe jsou (v paměti i na disku) uloženy hodnoty ze stejného sloupce
- Vertica, MonetDB, Netezza, ...
- ..., SQL Server, Oracle, ...

Sloupcové databáze

- Databáze pro datové sklady bývají často odlišně koncipovány (sloupcově orientované databáze)
- Vedle sebe jsou (v paměti i na disku) uloženy hodnoty ze stejného sloupce
- Vertica, MonetDB, Netezza, ...
- ..., SQL Server, Oracle, ...
- Rychlé vykonání agregačních funkcí nad mnoha řádky
- Komprese sloupce

Rozdělení dle datového modelu

- Dle použitého datového modelu a podporovaných operací:
  - Relační databáze (SQL)
  - Key-value, Dokumentové
  - XML databáze (XPath, XQuery)
  - Prostorové databáze (rozsahové dotazy)
  - Grafové databáze (kNN, hledání nejbližších sousedů)
- Důležité jsou zejména podporované operace
- Lehce můžeme například prostorová data uložit v relační tabulce, ale vykonání mnoha typických operací pro prostorová data bude v SQL dost komplikované a neefektivní

Rozdělení dle datového modelu

- Dle použitého datového modelu a podporovaných operací:
  - Relační databáze (SQL)
  - Key-value, Dokumentové
  - XML databáze (XPath, XQuery)
  - Prostorové databáze (rozsahové dotazy)
  - Grafové databáze (kNN, hledání nejbližších sousedů)
- Důležité jsou zejména podporované operace
- Lehce můžeme například prostorová data uložit v relační tabulce, ale vykonání mnoha typických operací pro prostorová data bude v SQL dost komplikované a neefektivní

Key-value

- Databázový systém je založen na nejjednodušším datovém modelu:
  - Základním elementem databázového systému je (klíč, hodnota)
  - Podporováno je vyhledání/aktualizace/mazání hodnoty na základě klíče
- Key-value se většinou liší celou řadou aspektů:
  - Podporované datové typy
  - Škálovatelností
  - Podporou datových struktur
  - Podporované operace na klíčích/hodnotách
- Redis, Memcached, ...
- ..., RocksDB, LMDB

Key-value

- Databázový systém je založen na nejjednodušším datovém modelu:
  - Základním elementem databázového systému je (klíč, hodnota)
  - Podporováno je vyhledání/aktualizace/mazání hodnoty na základě klíče
- Key-value se většinou liší celou řadou aspektů:
  - Podporované datové typy
  - Škálovatelností
  - Podporou datových struktur
  - Podporované operace na klíčích/hodnotách
- Redis, Memcached, ...
- ..., RocksDB, LMDB

Dokumentové

- Jedná se také o určitý typ key-value
- Nicméně hodnota je většinou JSON (popřípadě jiný semistrukturovaný formát)
- Jedna dvojice (klíč, JSON) je nazývána dokument
- Dokumenty jsou organizovány do kolekcí, což kromě zpřehlednění (kolekce analogii tabulky) obvykle přináší i výkonové zlepšení
- JSON je schema-less
- Možnost indexování hodnot v JSON (tzn. vyhledáváme dokumenty nejen dle klíče)
- MongoDB, CouchBase

XML databáze

- V XML databázích opět najdeme určité rysy předchozího datového modelu
- Nicméně tady není žádný klíč pro přístup k semistrukturovaným datům (t.j. k XML)
- Mále kolekci XML a propráci využíváme jazyky jako je XQuery
- XBase, Saxon

Rozdělení dle škálovatelnosti

- V současné době se stávají velmi aktuálními databáze, které se zaměřují na:
  - vysokou dostupnost - data jsou replikována i v několika datových centrech, aby se předešlo nedostupnosti
  - škálovatelnost - možnost bezpracně nastavovat počet databázových serverů úměrně zátěži
- Využívá se distribuce dat na více serverů, které spolu komunikují pouze po síti (nemají žádné sdílené prostředky)
- Dostupnost a zejména škálovatelnost má v relačních databázích své limity
- Z tohoto důvodu vznikla celá skupina databází, která se na tyto parametry zaměřuje (NoSQL, NewSQL)

Rozdělení dle škálovatelnosti

- V současné době se stávají velmi aktuálními databáze, které se zaměřují na:
  - vysokou dostupnost - data jsou replikována i v několika datových centrech, aby se předešlo nedostupnosti
  - škálovatelnost - možnost bezpracně nastavovat počet databázových serverů úměrně zátěži
- Využívá se distribuce dat na více serverů, které spolu komunikují pouze po síti (nemají žádné sdílené prostředky)
- Dostupnost a zejména škálovatelnost má v relačních databázích své limity
- Z tohoto důvodu vznikla celá skupina databází, která se na tyto parametry zaměřuje (NoSQL, NewSQL)

Scale up

- Také se nazývá vertikální škálování (Scale up)
- Scale up znamená možnost navyšování výkonu přidáváním jader procesoru na uzlu
- Tzn. paralelizace v rámci jednoho uzlu
- Toto již dnes nabízí celá řada databázových systémů
- Nicméně ne všechny (např. LevelDB)

Scale out

- Také se nazývá horizontální škálování (Scale out) a vertikální škálování (Scale up)
- Scale out znamená možnost navyšování výkonu přidáváním nových serverů (uzlů)
- Scale out může být problémem pro relační databáze
- Scale out je limitováno tzv. CAP teorémem

Scale out - CAP teorém

- Máme tři vlastnosti systému, které ovlivňujeme, když provádíme scale out:
  - Konzistence dat - všichni vidí stejná data naposledy potvrzená data
  - Dostupnost - každý požadavek dostane nějakou konzistentní odpověď (nemusí jít o poslední konzistetní stav)
  - Tolerance k výpadkům - systém funguje i při přerušení spojení mezi uzly
- Teorém tvrdí, že není možné dosáhnout všech tří vlastností zároveň
- Relační databáze jsou založeny konzistenci, takže trpí buď dostupnost, nebo tolerance k výpadkům v sítí

Scale out - NoSQL

- Relační databáze jsou založeny konzistenci, takže trpí buď dostupnost, nebo tolerance k výpadkům v sítí
- NoSQL se nicméně vzdává některých požadavků na konzistenci
- Mnoho aplikací nevyžaduje, aby všichni viděli poslední konzistentní stav (počet shlédnutí, atp.)
- Příznačné je, že nepoužívají SQL ani relační model
- MongoDB, Redis, Voldemort, atp.

Scale out - NewSQL

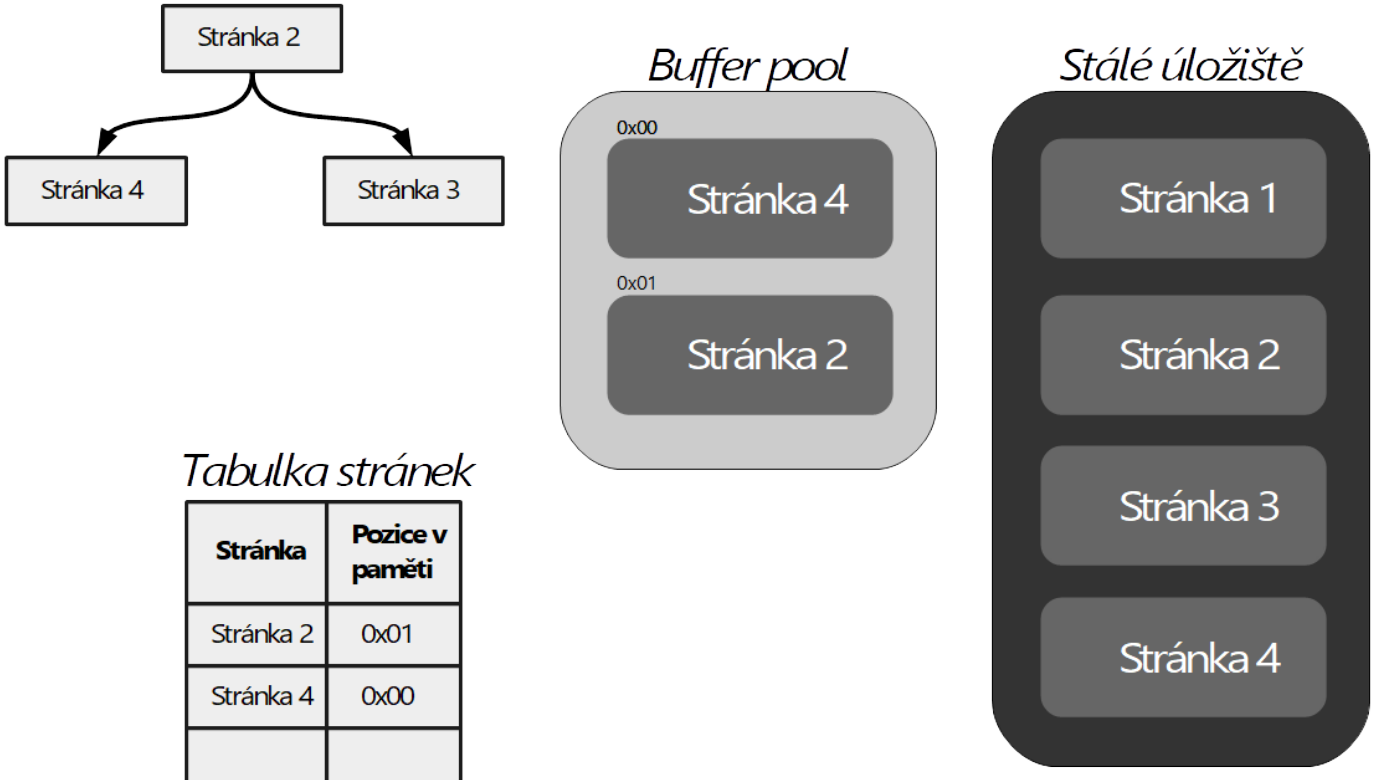
- NewSQL databázové systémy se snaží zvýšit požadavek na konzistenci dat na možné maximum při zachování:
  - Relačního datového modelu
  - SQL
- Obvykle se snaží i o zachování maxima z požadavků acid (ne vše je možné)
- VoltDB, MemSQL, ...

Rozdělení dle požadavků na paměť

- Tradiční databázové systémy byly navrženy s ohledem na klasickou architekturu Disk-Paměť-CPU
- Nicméně dnes cena pamětí neustále klesá
- Již nejsou vyjímkou servery se stovkami GB [1]

[1] https://www.brentozar.com/archive/2019/12/sql-constantcare-population-report-fall-2019/

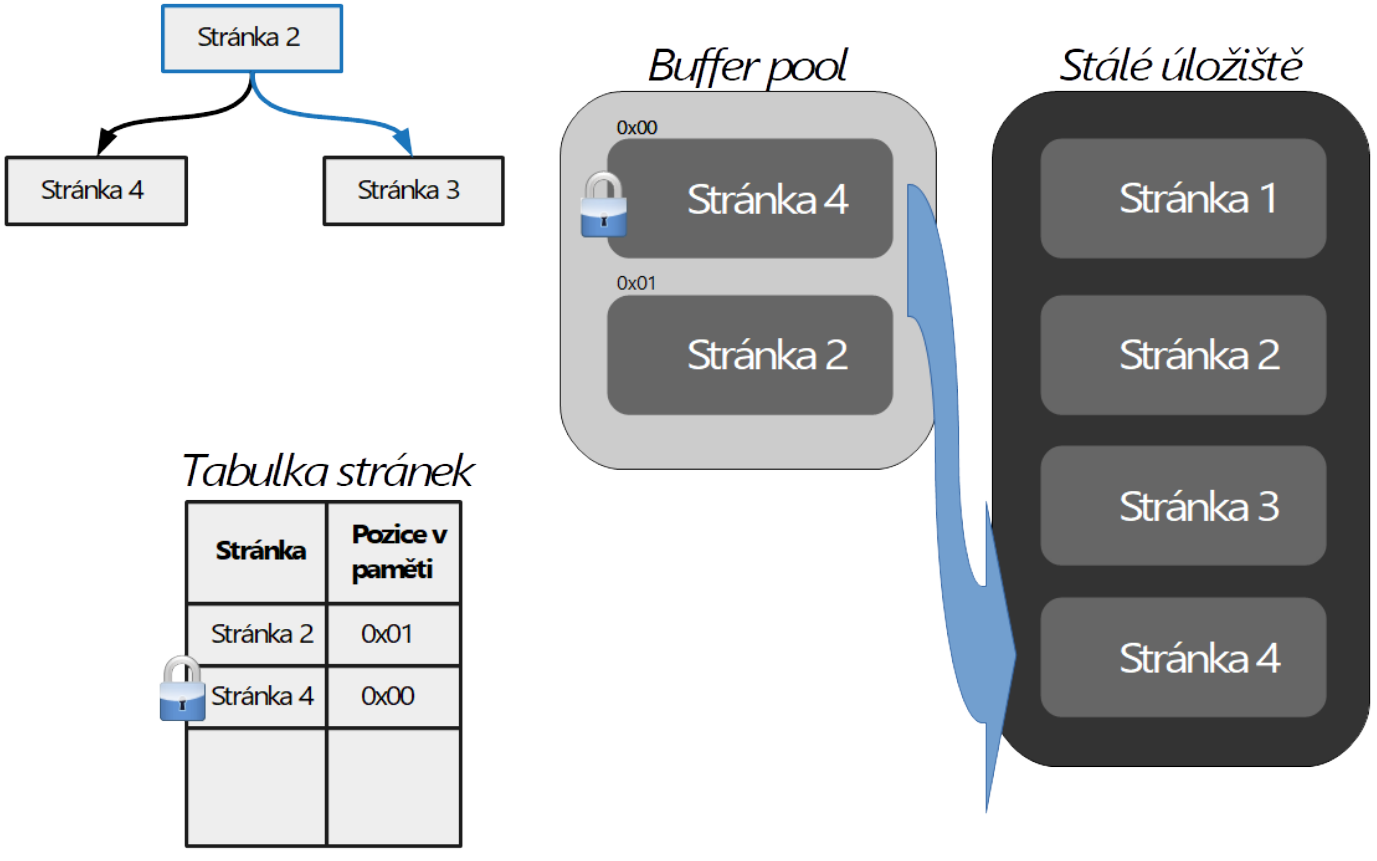
Diskově orientované systémy



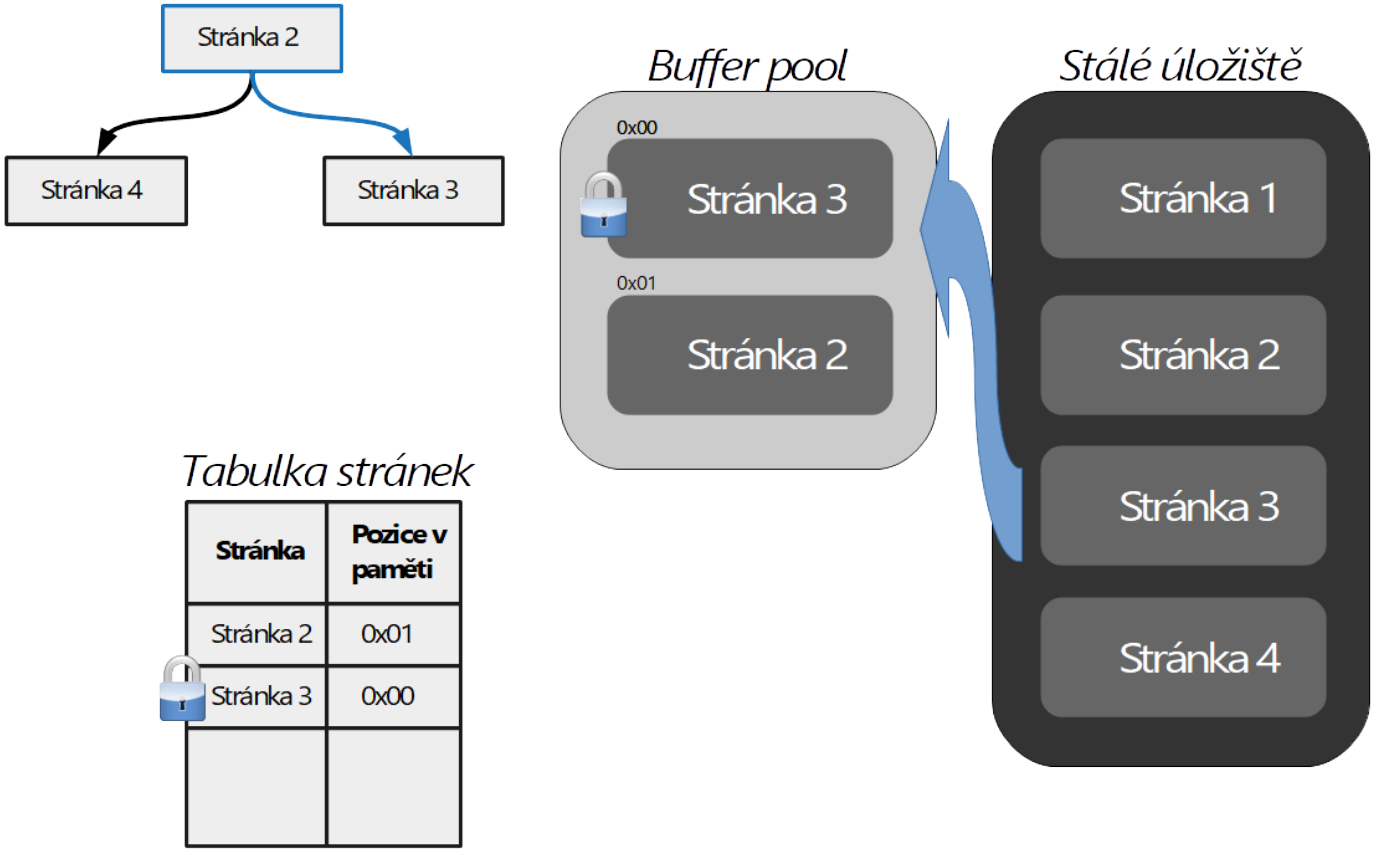
Tabulka stránek

Stránka	Pozice v paměti
Stránka 2	0x01
Stránka 4	0x00

Diskově orientované systémy



Diskově orientované systémy



Proč in-memory DBMS?

Návrh diskově orientovaných databázových systémů je založena na předpokladu, že se databáze nevleze do paměti, nicméně tento předpoklad u vězšiny dnešních databází neplatí [2]

[2] H. Stavros et al. Oltp through the looking glass, and what we found there, SIGMOD 2008

Proč in-memory DBMS?

Návrh diskově orientovaných databázových systémů je založena na předpokladu, že se databáze nevleze do paměti, nicméně tento předpoklad u vězšiny dnešních databází neplatí

In-memory databáze mohou vhodným návrhem dosáhnout až řádového zrychlení [3]

[3] T. J. Lehman et al. An evaluation of starburst? memory resident storage component, TKDE 1992

Database as a service

- Database as a service nabízí rychlé nasazení databáze bez nutnosti vlastního serveru
- Snižuje nutnost administrace
- Často umožňuje jednoduchou škálovatelnost
- Nicméně má řadu úskalí
  - Bezpečnost dat
  - Může překvapit i účet v případě špatného nastavení služby a nečekané chyby v návrhu ...
- DynamoDB, Amazon RDS, Google Spanner, SQL Azure, ...

Díky za pozornost