Fyzický pohled na data a vývoj aplikací nad databází

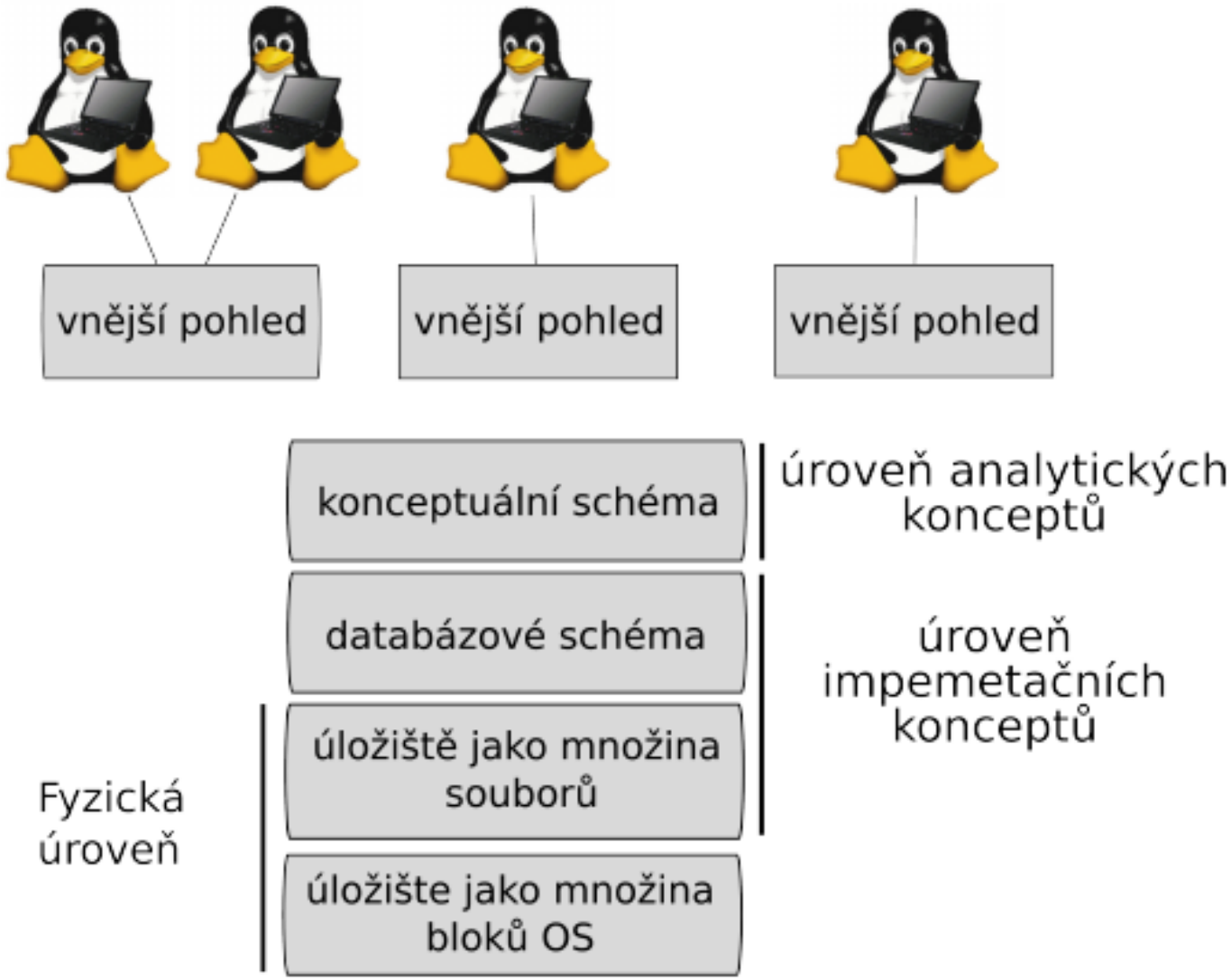
Michal Valenta

Katedra softwarového inženýrství
Fakulta informačních technologií
České vysoké učení technické v Praze

BI-DBS, LS 2021/2022

https://courses.fit.cvut.cz/BI-DBS/

Kontext: různé úrovně pohledu na data



Prezentace založena na RDBMS Oracle

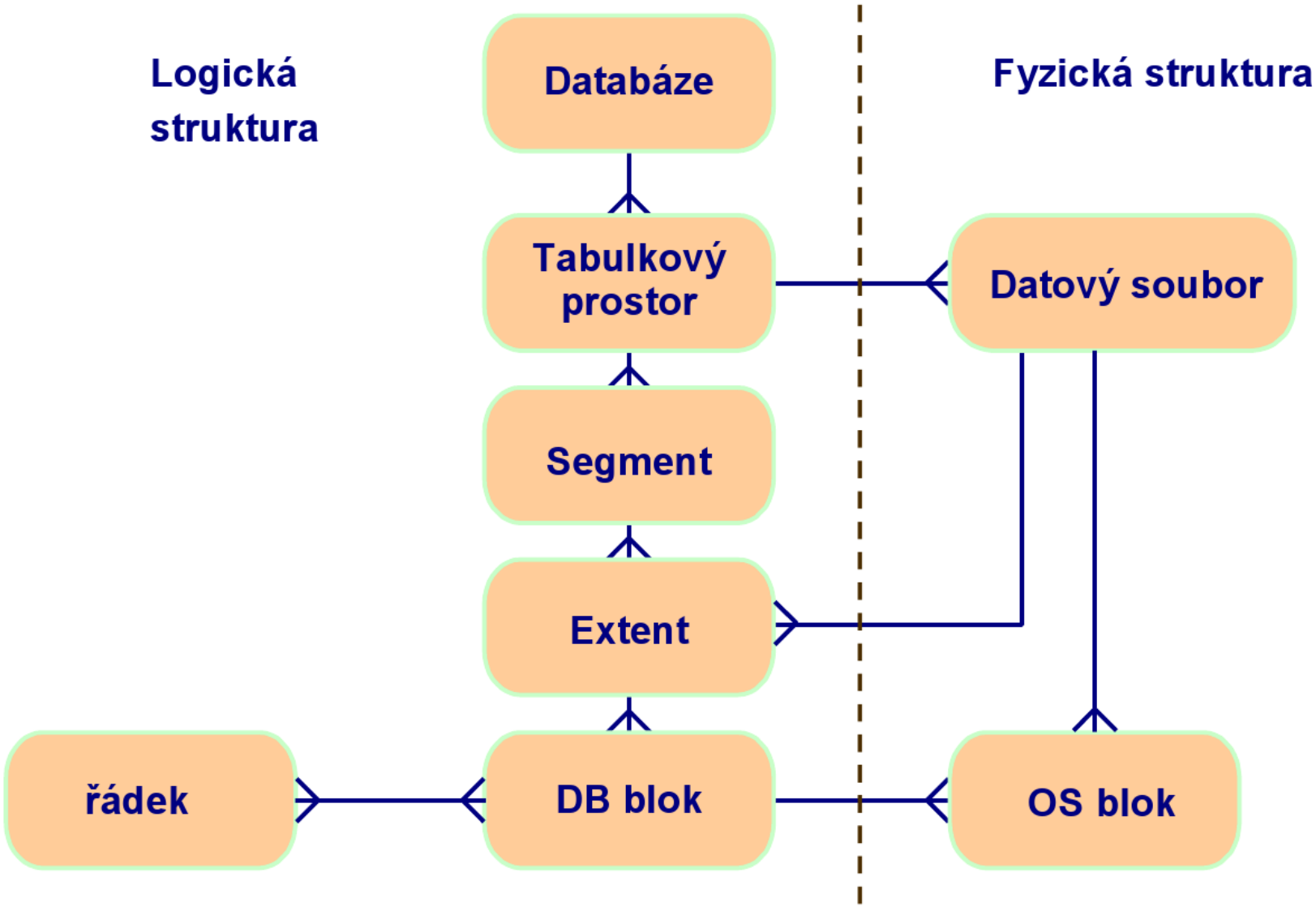
- **indexy – nejdůležitější partie této přednášky** – jsou ale implementovány ve všech relačních DBMS velmi podobně
- naopak: IOT a cluster všude nenajdete
- **koncept ROWID** je z Oracle, ale jinde najdete obdobu
- na úrovni architektury je mnoho rozdílů:
  - Oracle: user a schéma jsou totožné
  - Oracle: vytvoření databáze má velkou režii a sebere hodně zdrojů, typicky: jedna databáze – více aplikací
  - PostgreSQL: user je na úrovni clusteru, v databázi lze mít mnoho schémat (obdoba namespace)
  - PostgreSQL, MySQL: vytvoření databáze má velmi malou režii a spotřebu zdrojů, typicky: jedna aplikace – jedna databáze
  - PostgreSQL, Oracle: jednotný způsob fyzického uložení dat (vzájemně velmi odlišné)
  - MySQL (MariaDB): různé „storage engines“ na úrovni tabulek
- Ale indexy jsou potřeba všude!

Motivace – užitečnost indexů

Praktická ukázka

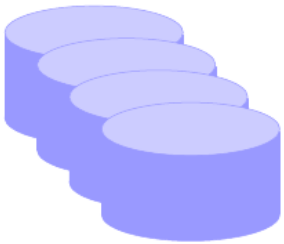
- prováděcí plán
- statistiky objektů
- příklad SELECTu s indexem a bez indexu

Oracle – fyzická a logická struktura databáze

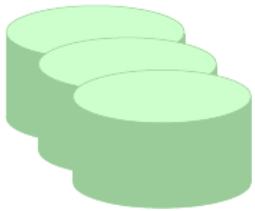


Oracle – fyzická struktura databáze

Řídicí soubor a jeho zrcadlení

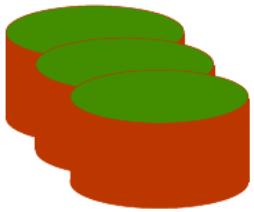


Datové soubory



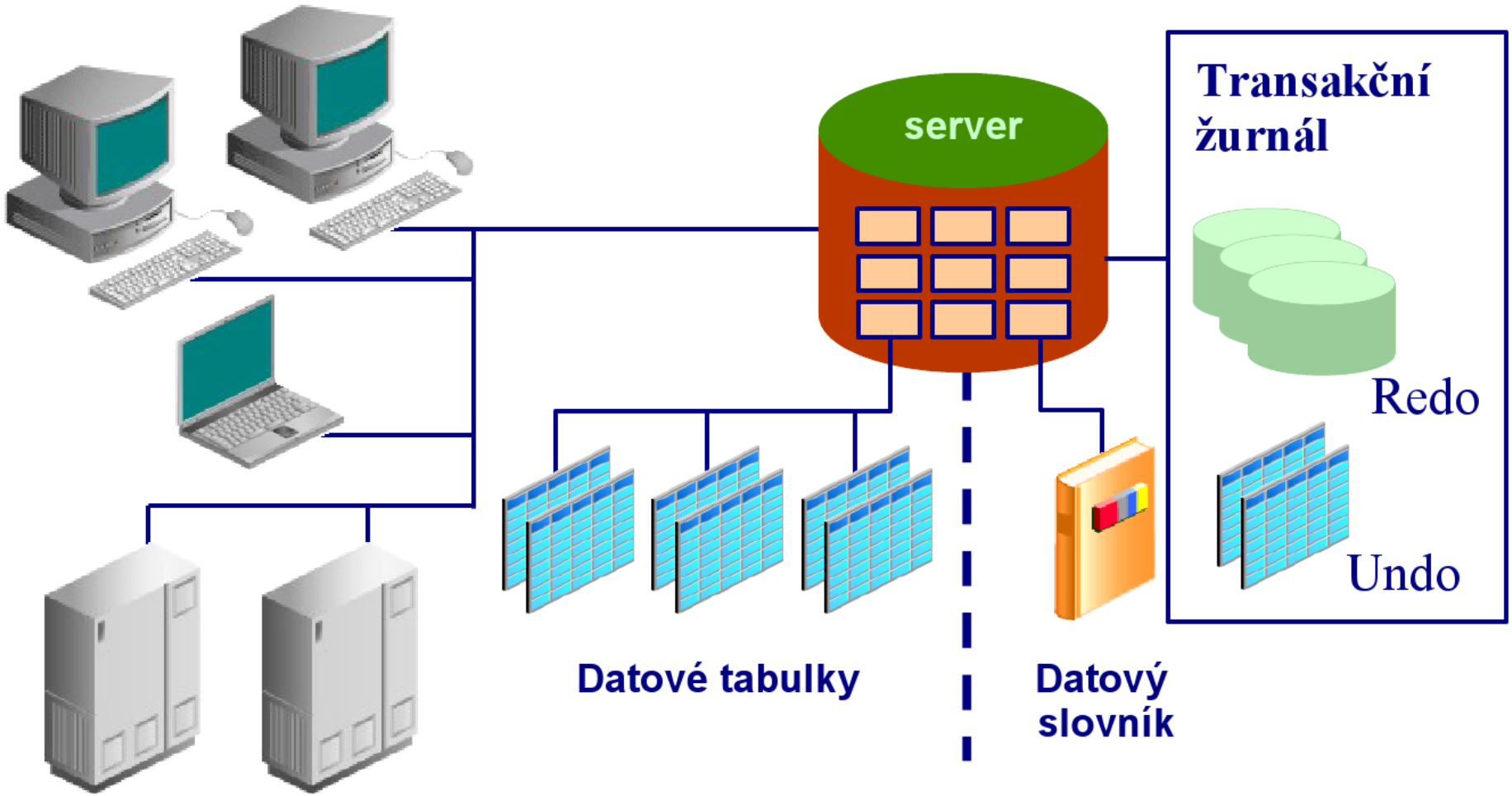
Online redo log soubory a jejich zrcadlení

Parametry

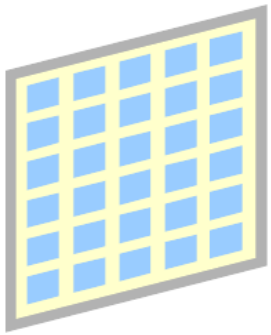


Archivní kopie redo log souborů

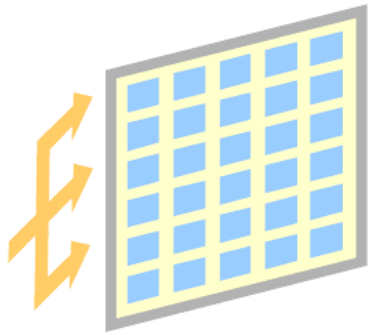
Co všechno kromě (aplikačních) dat patří k databázi



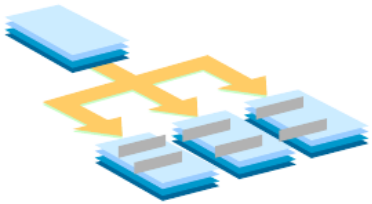
Oracle – způsoby uložení (relační) tabulky



HEAP tabulka



HEAP tabulka s indexy



Tabulka s indexovou organizací

tabulka ve shluku (cluster)

Unikátní adresa/identifikátor řádku – Oracle ROWID

- ROWID nebo jiný unikátní identifikátor řádku (v jiných RDBMS) se využívá (mimo jiné) pro implementaci indexů. Je to „pseudosloupec“ a „implementační trik“.
- Primární (a unikátní) klíče z vlastních atributů tabulek jsou stále potřebné – dobrý návrh, bezpečná práce s databází!
- ROWID přímo odkazuje k fyzickému umístění řádky; struktura: <object_id, file_id, block_id, row_id>
- ROWID se během života databáze může měnit, nepoužívat v aplikacích!

```
select rowid, t.* from Titul t
ROWID                  TITUL_ID  NAZEV              ROK_VYROBY
---------------------- --------- ------------------ ----------
AAAsaTAAHAAA656AAA           1   Název_titulu _1     01.01.2005
AAAsaTAAHAAA656AAB           2   Název_titulu _2     01.01.2005
AAAsaTAAHAAA656AAC           3   Název_titulu _3     01.01.2005
. . .
```

Asociativní výběr – „prohrabání hromady“

```
select * from Titul t where NAZEV='Název_titulu _2'
```

Adresní výběr (nadrelační rys):

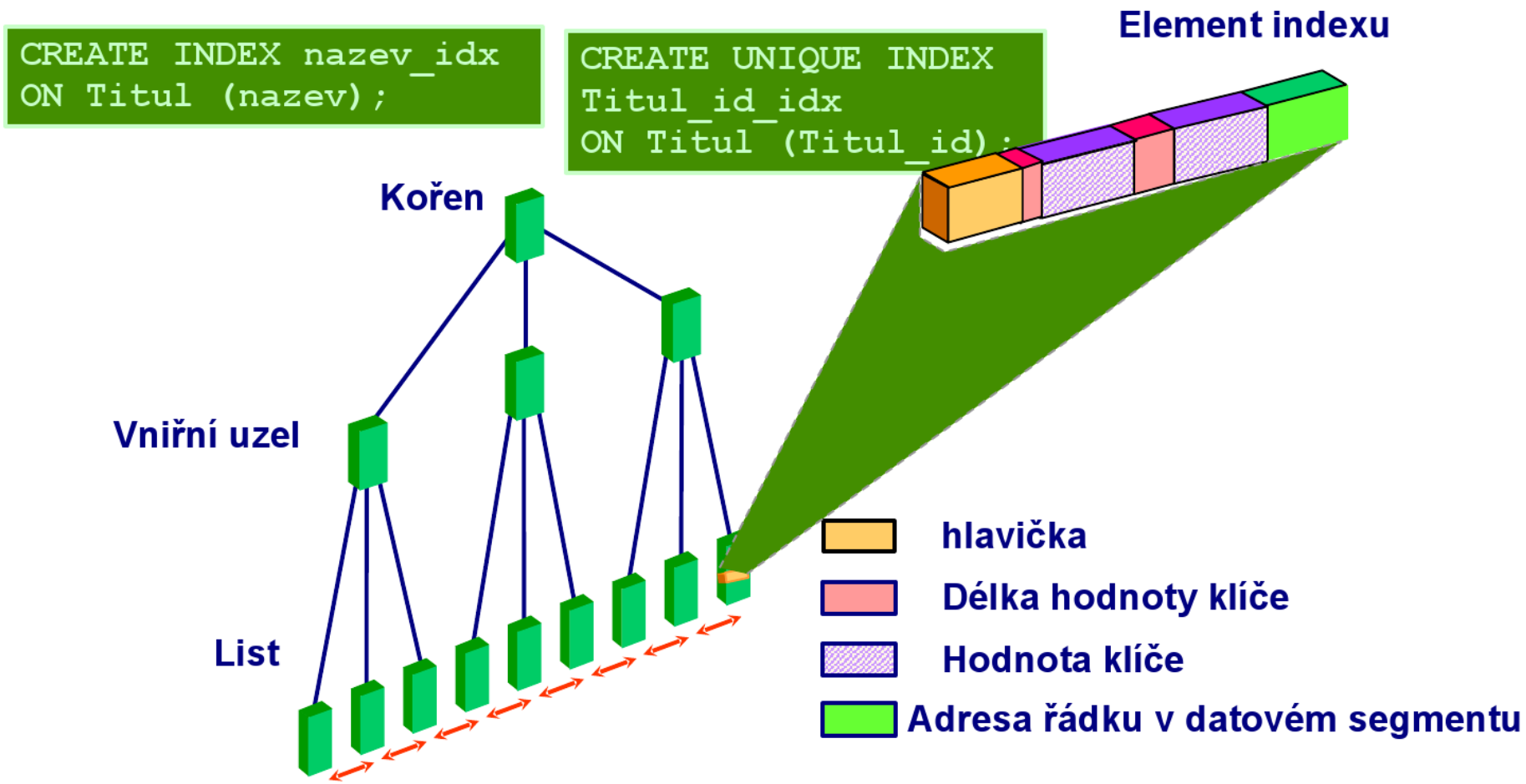
```
select * from Titul t where ROWID ='AAAsaTAAHAAA656AAB'
```

Indexy v relačních databázích

- Pomocná (sekundární) struktura – rychlý přístup k datům (řádkům) podle klíče indexu (index key).
- Nad jednou tabulkou může existovat mnoho různých indexů (každý pro jiný klíč).
- Typů indexů je více, vyžadujeme znalost dvou konceptů:
  - indexy založené na struktuře B-stromu (B*-Tree based indexes),
  - bitmapové indexy.
- Selektivita (kardinalita) klíče indexu:
  #různých_hodnot / #řádků_tabulky (nemusí být unikátní)
  - vysoká selektivita (blízko 1): B-stromové indexy
    (PK, UK, (username), (prijmeni, jmeno), ...)
  - velmi nízká selektivita: bitmapové indexy
    ((pohlaví), (okres), (vzdělání), ...)
- DML – aktualizace dotčených indexů;
  zrychlíme SELECTy, zpomalíme DML.
- Pro PK a UK indexy vytvořeny automaticky
  efektivní kontrola platnosti IO.

Index typu B*-Tree

CREATE INDEX nazev_idx on Titul (nazev);
CREATE UINIQUE INDEX titul_id_idx on Titul (titul_id);



Index typu B-Tree - charakteristika

- faktor větvení (v praxi cca 100), hloubka stromu (v praxi menší než 4)
- důležité je vyvážení stromu, na pozadí, provádí DBMS, rekurzivní
- níže uvedené jen pro ilustraci (detaily BI-AG1), zde nevyžadujeme

B - strom (řádu m) je m-ární strom, splňující následující omezení

- Kořen má nejméně 2 potomky, pokud není listem
- každý uzel kromě kořene a listu má nejméně [m/2] a nejvýše m potomků
- každý uzel ma nejméně $\lceil m/2 \rceil - 1$ a nejvíce $m - 1$ datových záznamů (většinou jen klíčů)
- všechny cesty ve stromě jsou stejně dlouhé
- data v nelistovém ulzu jsou organizována

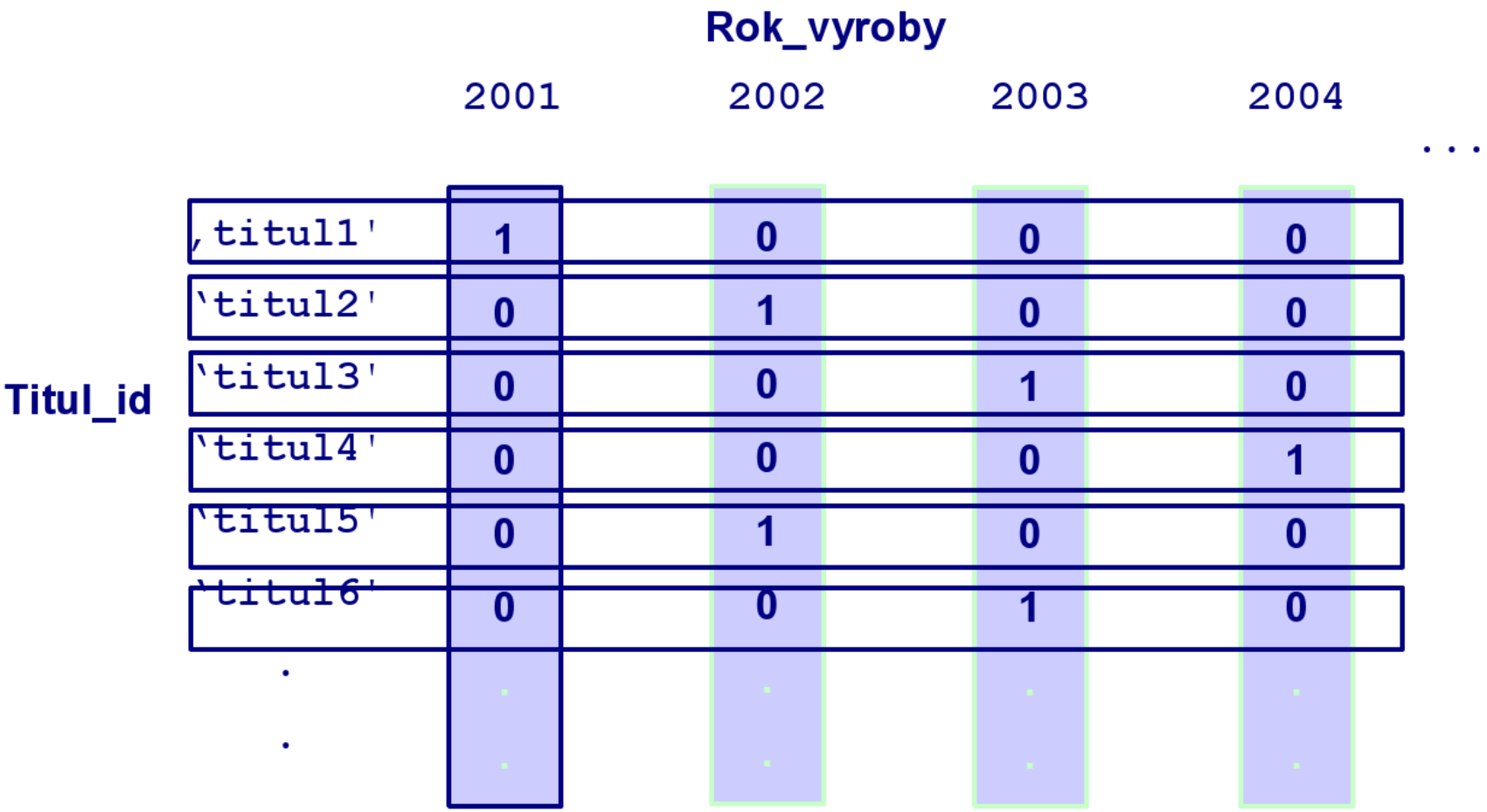
  $p_0(k_1, p_1), (k_2, p_2), ..., (k_n, p_n), u$

  - kde $p_0, p_1, ..., p_n$ jsou ukazatele na potomky
  - $k_0, k_1, ..., k_n$ jsou klíče
  - u je nevyužitý prostor
  - záznamy(k_i, p_i) jsou uspořádany vzestupně podle klíčů
  - $\lceil m/2 \rceil - 1 <= n <= m - 1$

- odpovídá-li ukazateli p_i, kde $i \in < 1, n >$ podstrom $U(p_i)$ platí:
  - (i) pro každé $k \in U(p_{i-1})$ je $k <= k_i$
  - (ii) pro každé $k \in U(p_i)$ je $k > k_i$
- listy obsahují úplnou množinu klíčů a mohou se lišit strukturou.

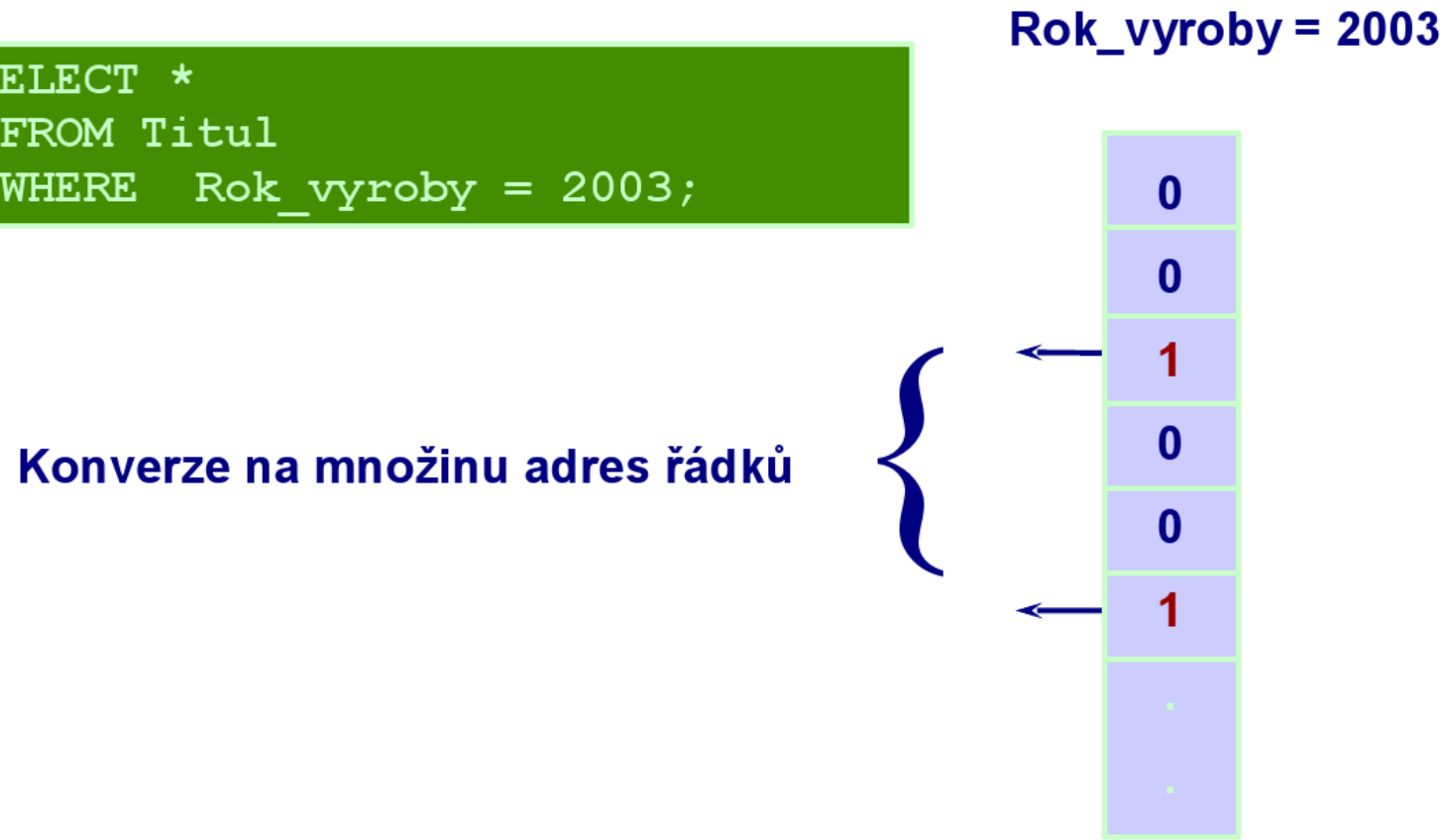
Bitmapové indexy

```
CREATE BITMAP INDEX rok_id_bix
ON Titul (rok_vyroby);
```



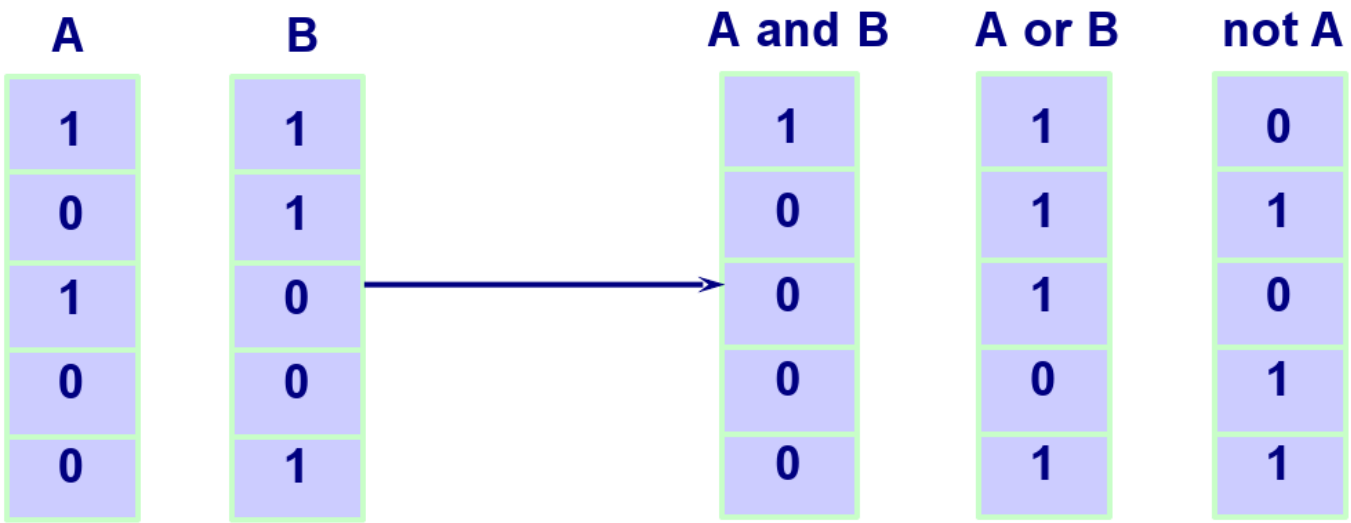
Použití bitmapového indexu při vyhodnocení dotazu

```
SELECT *
 FROM Titul
 WHERE  Rok_vyroby = 2003;
```



Použití bitmapového indexu při vyhodnocení dotazu

- Výběrová podmínka s operátorem IN
- Výberová podmínka s operátory AND/OR/NOT
- Složité výběrové podmínky (WHERE) kombinované z výše uvedených
- Obvykle se využívají spíše v datových skladech

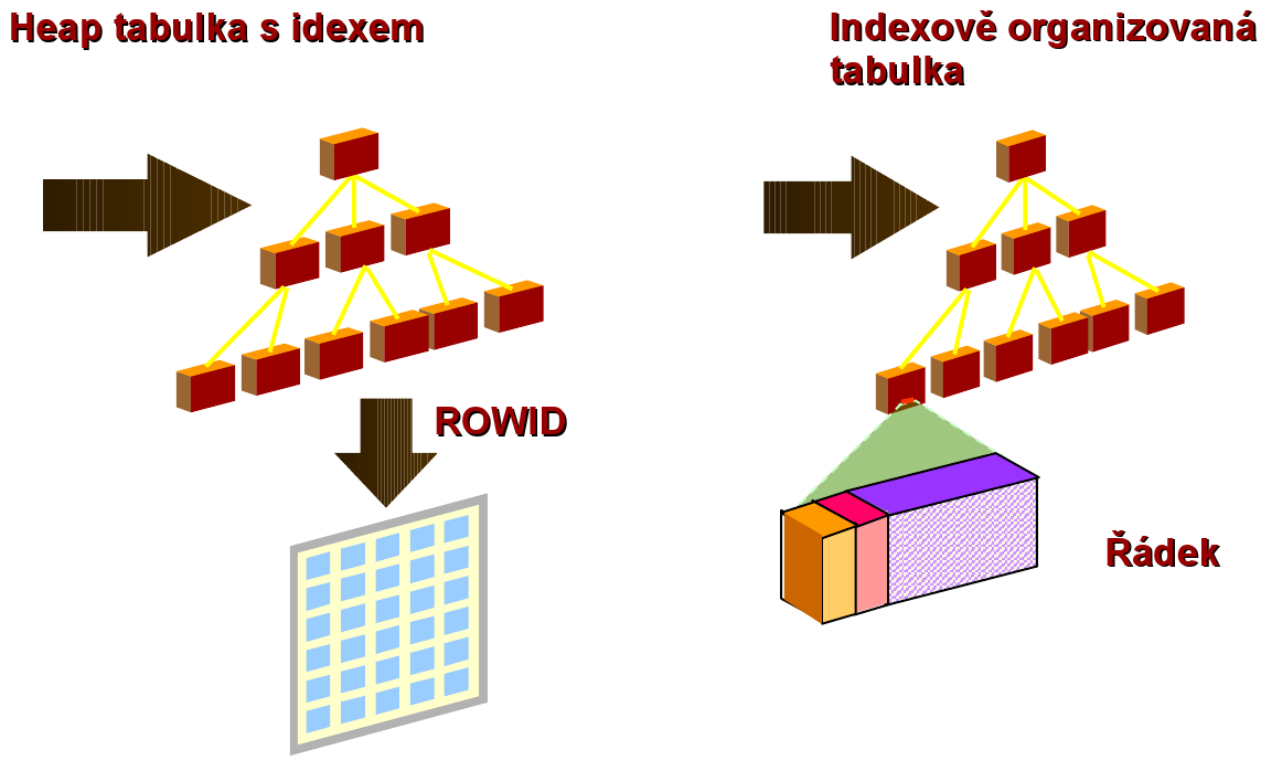


Porovnání indexů typu B-strom a bitmapa

B-strom	Bitová mapa
Sloupce s vysokou kardinalitou	Sloupce s nízkou kardinalitou
DML operace relativně drahé	DML operace velmi drahé
Vhodné pro OLTP	Vhodné pro ad hoc dotazy v datových skladech DSS

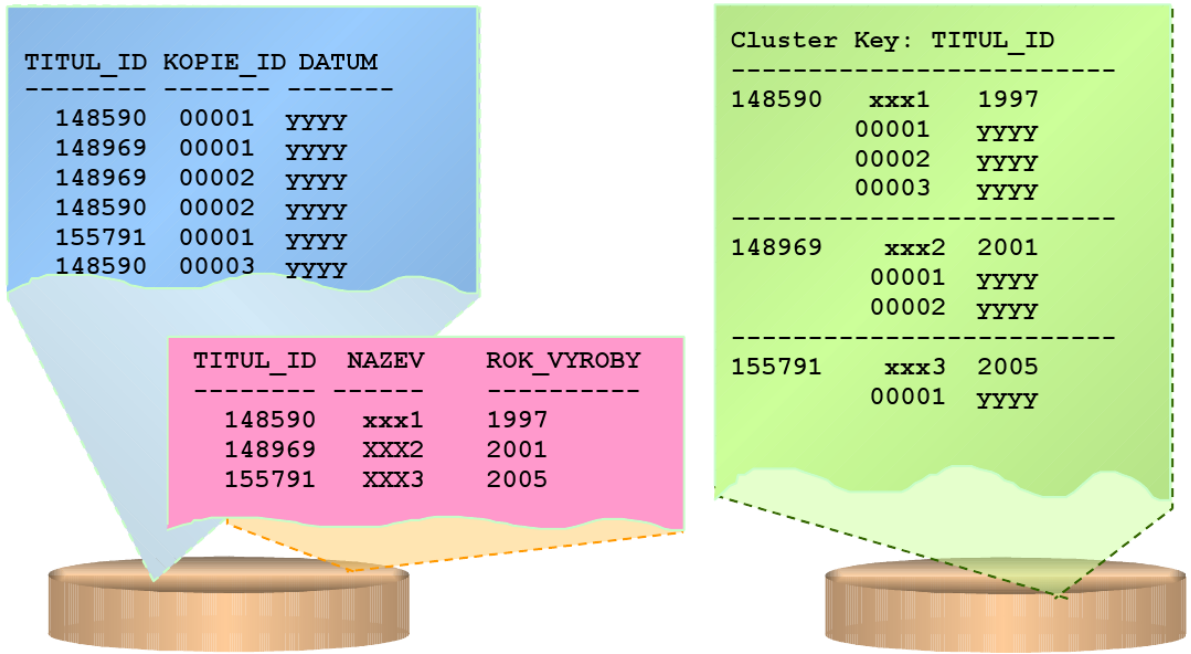
- OLTP – OnLine Transaction Processing
- DSS – Decision Support Systems;
  dnes spíše termín OLAP (OnLine Analytical Processing)
- typická zatížení databáze, přizpůsobuje se jim konfigurace DB

Indexově organizovaná tabulka (Index-organized table)



Ještě rychlejší přístup přes klíč indexu (ale je jen jeden takový index).

Shluk (Cluster)



Samostatné tabulky s vazbou přes cizí klíč

Tabulky ve shluku

Join tabulek, je hotový i ve fyzickém uložení.
Ale co s různými počty záznamů ve skupině (bucketu)?

Další techniky zrychlení přístupu k datům

Používané techniky:

- sledování frekventovaných SQL příkazů a jejich optimalizace (indexy a spol.)
- speciální struktury pro uložení dat
- materializované pohledy,
- zavedení redundance (denormalizace),
- partitions (partitioned tables),
- distribuované databáze, ...

Možnosti ladění jednoho DB serveru:

- systémové zdroje (paměť - velikost a struktura)
- konfigurace serveru (disky, parametry databáze, zálohování)

Aplikace nad databázemi

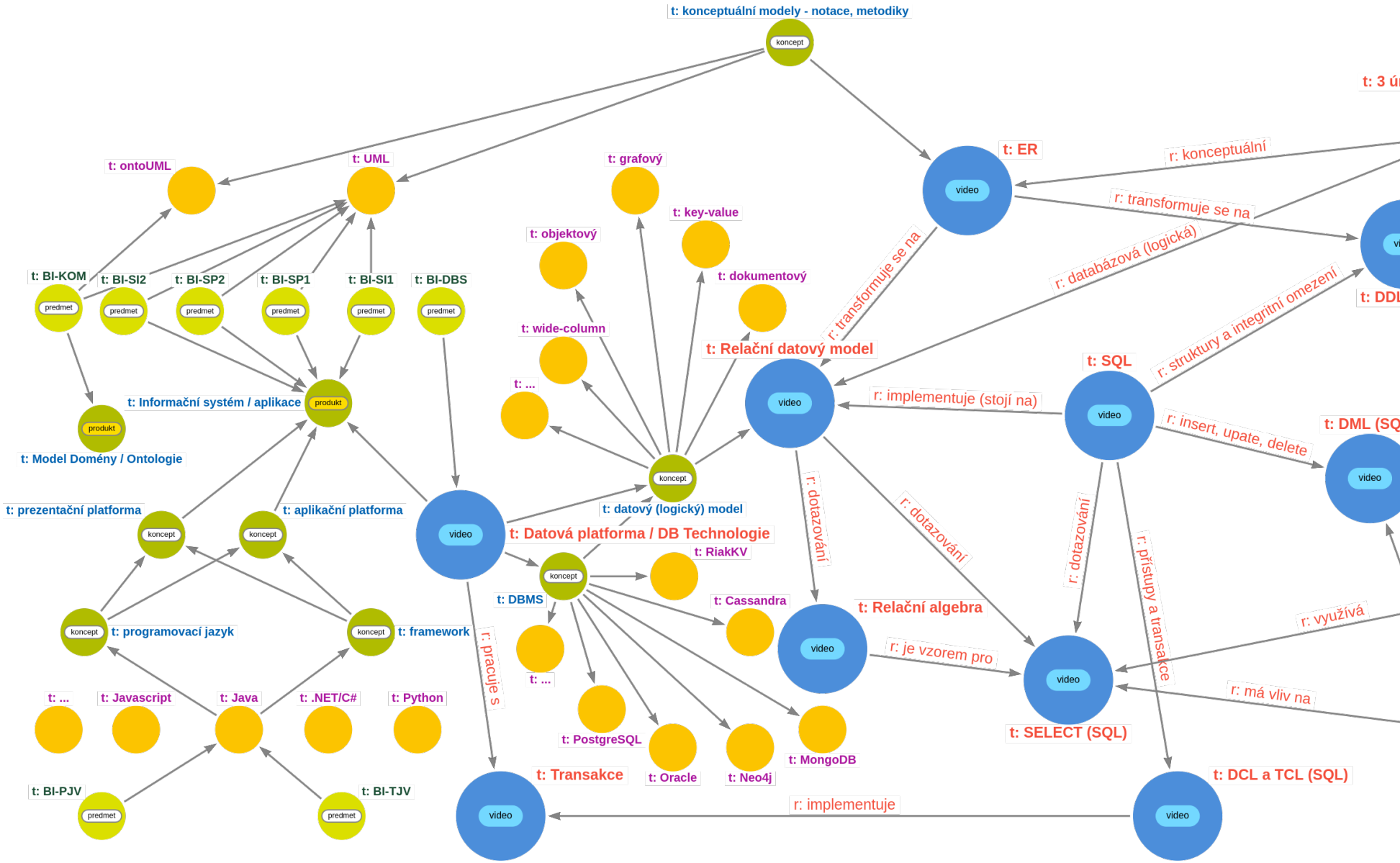
- dnes typicky třívrstvé aplikace
  - datová vrstva
    PostgreSQL, MariaDB, Oracle, MS SQL, MongoDB, Neo4j, ...)
  - aplikační logika
    PJ+framework – Java+Spring, Javascript+NodeJS, ...
  - prezentační vrstva (user interface)
    PJ+frameworky/knihovny – Javascript+ReactJS, Javascript+Vue,...
- rozhraní mezi aplikační a prezentační vrstvou
  - REST API
  - SOAP
  - GraphQL, microservices, ...
- vývojáři (typicky)
  - backend
    obvykle REST API (nebo jiné API, viz výše)
  - frontend vývojář
    vstupem je API, výstupem (responzivní) www aplikace
  - full-stack vývojář

Vyzkoušíte si, budete-li studovat SI/WI/PG v BI-TJV nebo BI-TWA, dále v BI-SP1/BI-SP2 a nejspíš i v bakalářské práci.

Aplikace nad databázemi – příklady ze školního prostředí

- Portál DBS (PostgreSQL + PHP/Nette)
- Anketa ČVUT (Oracle + Spring + ReactJS)
- Knihovna moderní české poezie (MongoDB + NodeJS + ReactJS)

Vývoj aplikací ve studiu SI – kontext



SQL z „normálního“ programovacího jazyka

- fáze zpracování dotazu
  - parse
  - bind
  - execute
  - fetch
  - + struktura pro návrat chybového kódu
- API kopíruje fáze zpracování SQL dotazu
- specifické knihovny (PHP, Python, C++, ...)
- standardizované API a drivery (ODBC, JDBC, ...)

... a přístupy, které odstiňují používání databáze:

- ORM – Object-Relational Mapping (objetkově-relační mapování)
  projekty: Spring ORM, hibernate, tapestry,...

Proč ORM?

„Semantic gap“: relační data vs. objektové aplikace.

K zapamatování – fyzické uložení dat

- (relační) databáze bez indexů nefungují rozumně – indexy jsou nutné (pro větší data)
- DB stroj často některé indexy vytváří automaticky kvůli kontrole IO (PK, UK)
- v OLTP se nejčastěji používají indexy na bázi B-stromů (tam, kde jsou data (skoro) unikátní)
- kde jsou data velmi neunikátní (např. pohlaví) a potřebuji indexovat, tam se používají bitmapové indexy
- indexy je třeba udržovat (zjednodušeně – ušetřím na dotazech, platím více při DML)
- klíč indexu (indexované atributy) může být složený / jednoduchý
- index může být unikátní / neunikátní.

K zapamatování – vývoj aplikací nad DB

- v BI-DBS jste se naučili:
  - navrhnout databázi na konceptuální úrovni (notace E-R)
  - převést E-R schéma na relační
  - pracovat v relační databázi (SQL)
  - koncepty: transakce, normalizace
- pokračování (zejména pro vývoj aplikací (SI/WI/PG/ISM)
  - postupy/přístupy k návrhu/realizaci/rozvoji SW systémů: BI-SWI, BI-KOM
  - technologie: BI-IDO, BI-TJV, BI-TWA
  - praxe, týmový vývoji: BI-SP1, BI-SP2
  - programovací jazyky: BI-PYT, BI-PJV, BI-PCS, BI-PHP, BI-PJS, ...
  - databáze: BI-SQL, BI-BIG