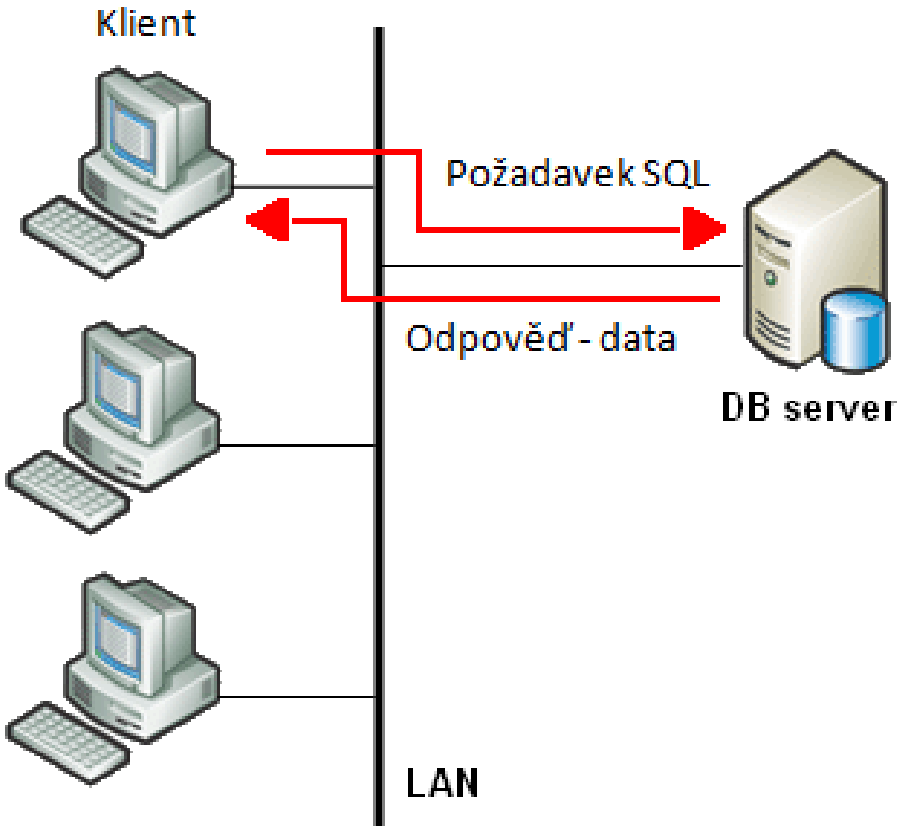
Distribuované databázové systémy

Luděk Kratochvíl
CCA Group a.s.
ludek.kratochvil@cca.cz

Agenda

- Přehled architektur databázových systémů
- DDBMS - teoreticky
- Jak to dělá Oracle
- Případové studie

Klient-Server



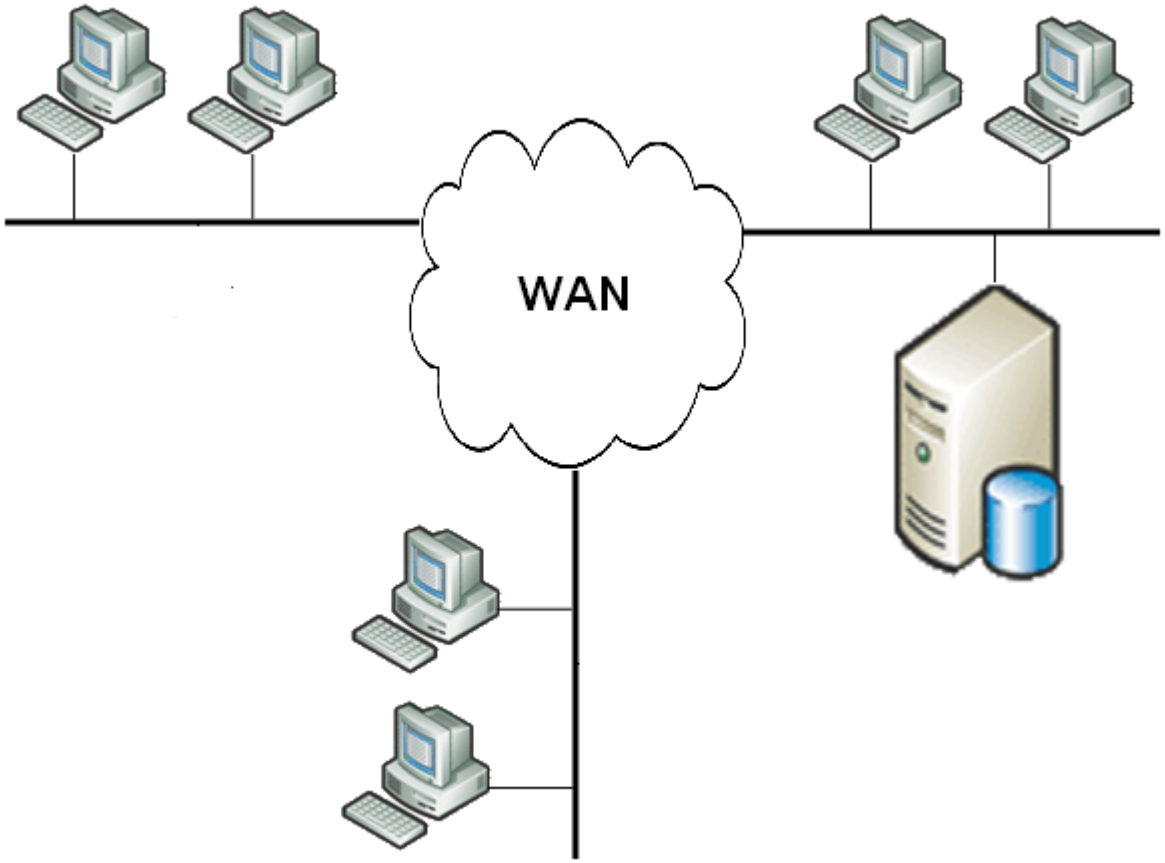
Zpracování dat je rozděleno mezi dvě části DBMS:

- na klientovi je tzv. databázový klient (plus uživatelská aplikace, která ho využívá)
- na HW serveru je hlavní část DBMS, databázový server

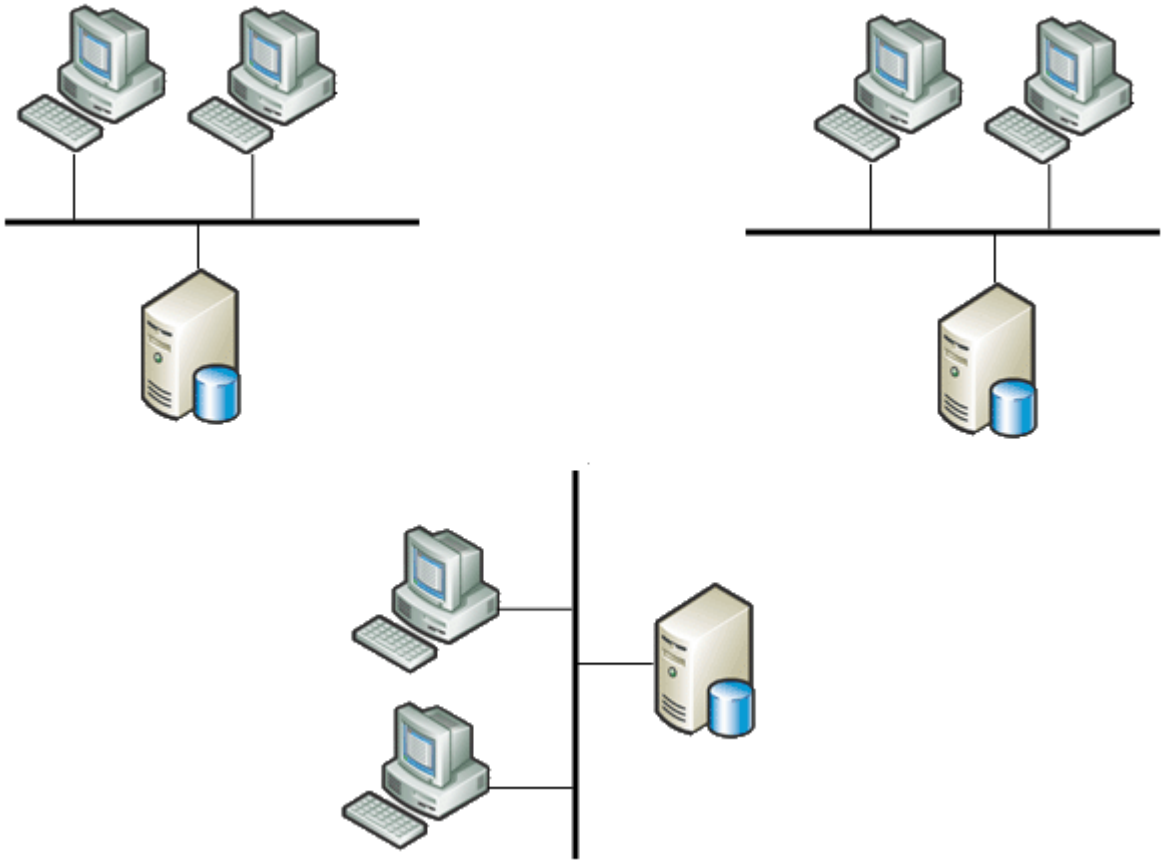
Protokol mezi klientem a serverem je pro relační databáze typicky SQL (Oracle Net8)


Česká republika
(Czech Republic)
Dresden
Chemnitz
Zwickau
Jelenia Góra
Świdnica
Walbrzych
Opole
Most
Karlovy Vary
Kladno
Příbram
Tábor
Písek
Strakonice
České Budějovice
Jihlava
Třebíč
Blansko
Brno
Kroměříž
Šumperk
Znojmo
Hodonín
Mladá
E55
E49
E67
E50
E442

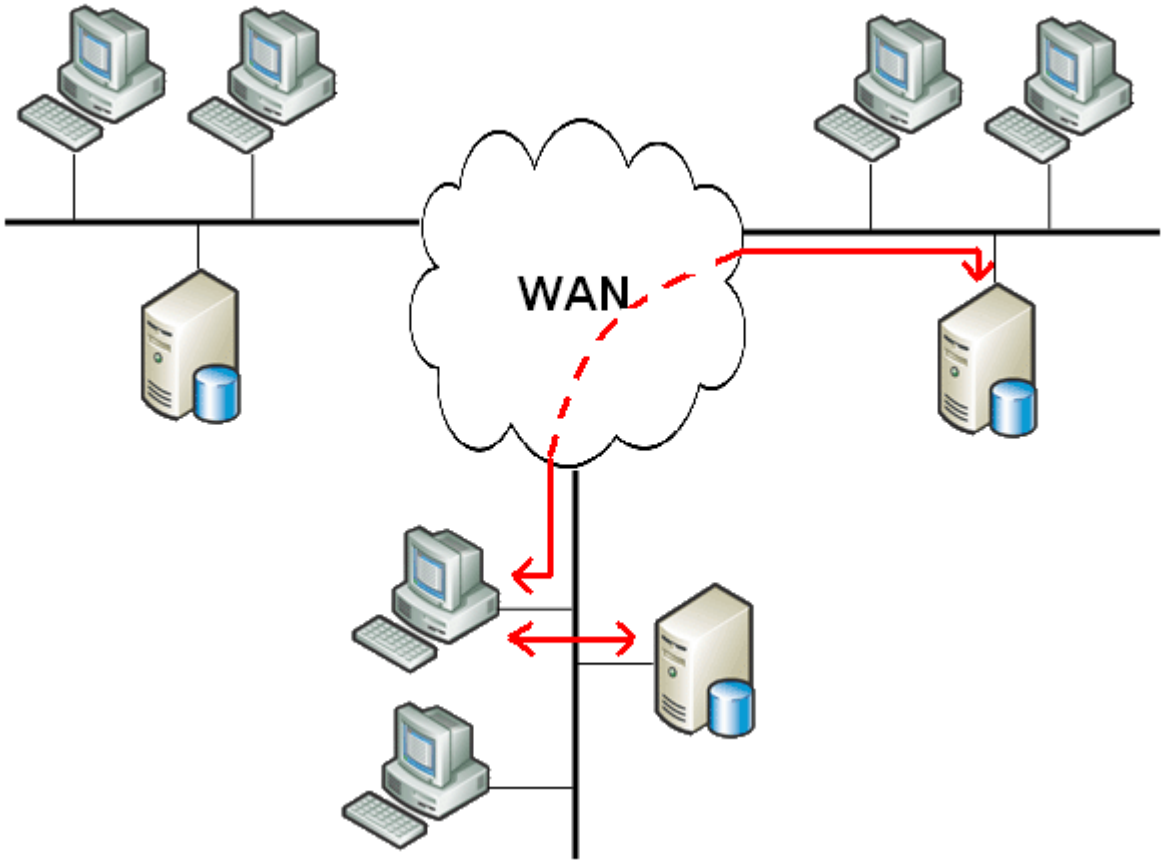
Centrální DBS



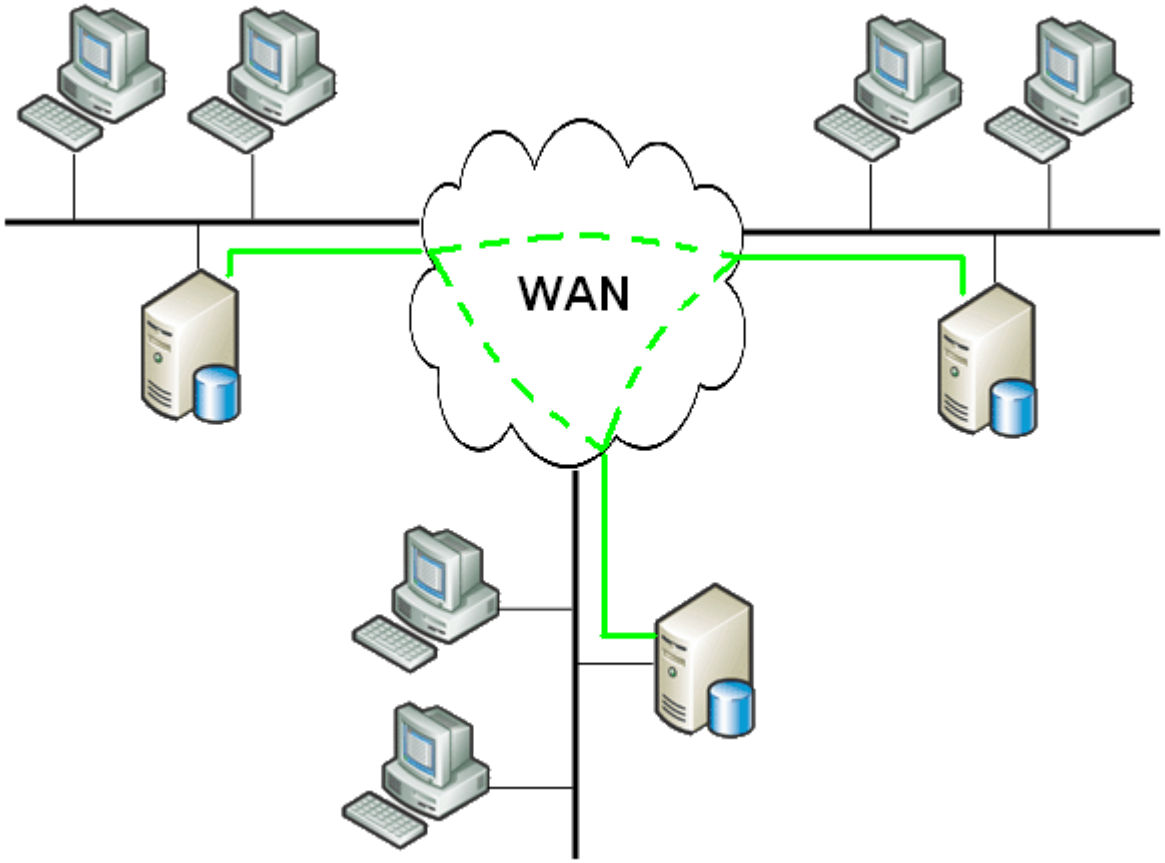
Decentralizovaný DBS



Decentralizovaný DBS



Federativní DBS



DDBMS

- Kolekce DBMS propojených sítí (logicky tvoří jeden systém)
- Lokální data jsou řízena lokálním DBMS
- DBMS zajišťuje globální zpracování (práci s daty celé distribuované databáze)
- Integrace dat bez nežádoucí centralizace
- Návrh:
  - zdola-nahoru – integrace existujících systémů
  - shora-dolů – začíná se na „zelené louce“, návrh datového modelu, návrh distribuce dat

Výhody DDBMS

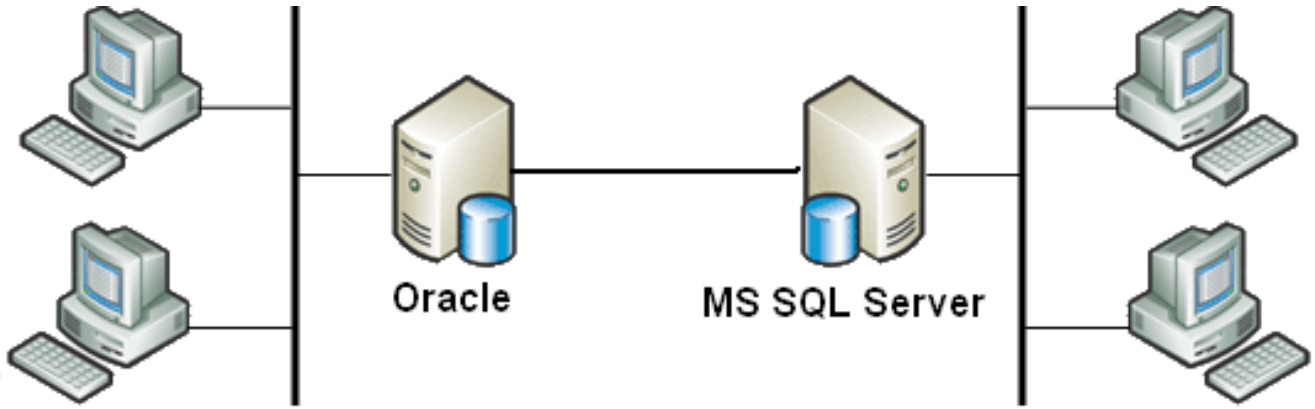
- Reflektuje organizační strukturu
- Data mohou být uložena blízko místa nejčastějšího používání
- Oproti centrálnímu řešení zvyšuje výkon
  - Práce s lokálními daty je rychlá (lokální zpracování)
  - Skrytý paralelismus při zpracování SQL
- Lze pracovat i s daty jiných uzlů (virtuální databáze)
- Zvyšuje spolehlivost – výpadek jednoho uzlu nezastaví celý systém
- Umožňuje integraci existujících systémů

Nevýhody DDBMS

- Složitost DDBMS
- Bezpečnost (nutno zabezpečit více uzlů + síťové přenosy)
- Náročná kontrola integrity dat (PK, FK) – jen v rámci uzlu
- Neexistence standardů (na rozdíl třeba od SQL)
- Nedostatek zkušeností
- Složitý návrh databáze

Klasifikace DDBMS

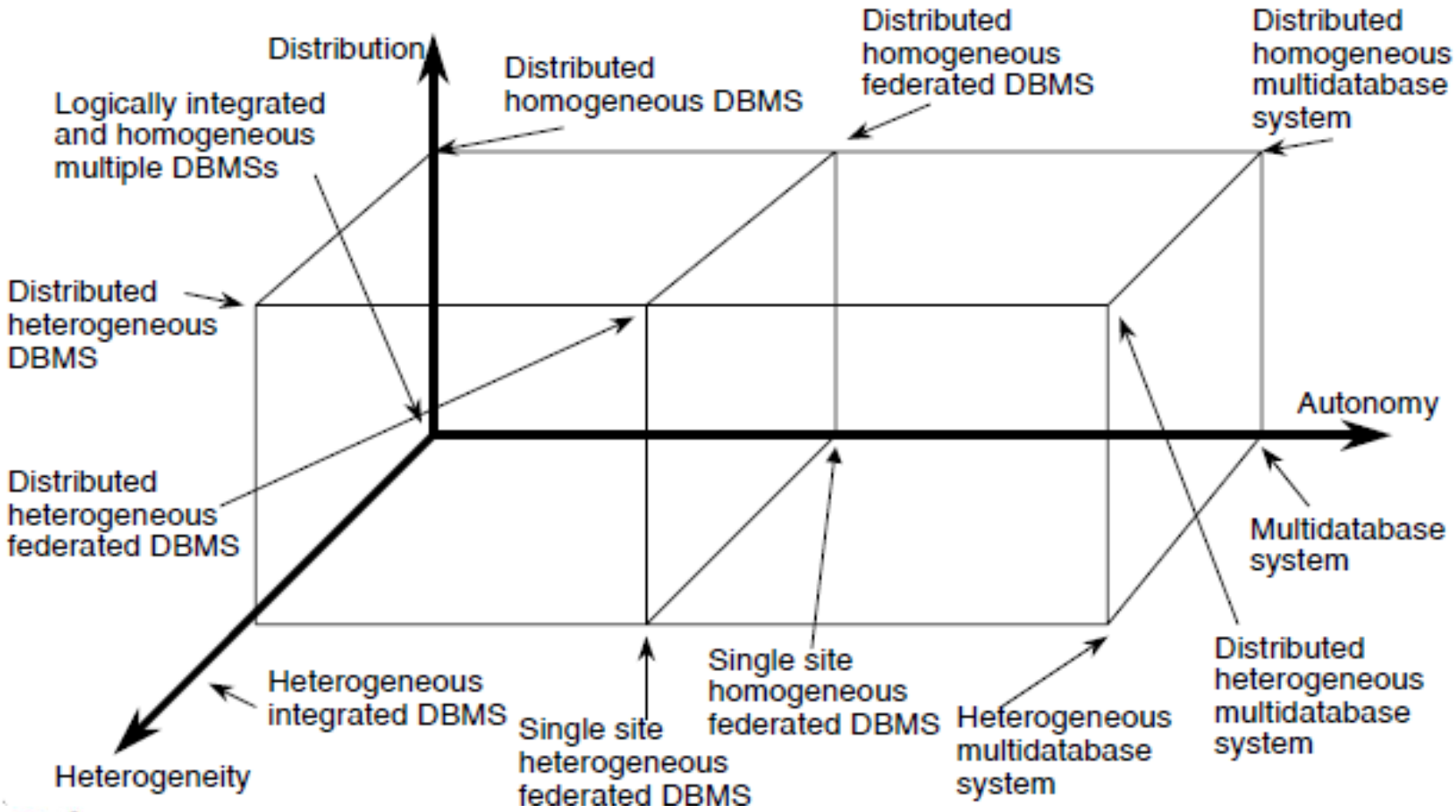
- Homogenní DDBMS – v každém uzlu je stejný DBMS
- Heterogenní DDBMS - v různých uzlech různé DBMS (různí výrobci, různé datové modely)



Klasifikace DDBMS

- Stupeň autonomie jednotlivých uzlů:
  - Plně autonomní (izolované, neví o sobě)
  - Částečně autonomní = Federativní (lokální data řízena plně lokálně, možno zpřístupnit vybraná data pro distribuované zpracování, možno přistupovat na data z ostatních databází)
  - Těsná integrace – pro uživatele v podstatě jeden DBMS (vše se zpracovává globálně)

Klasifikace DDBMS



DDBMS

- Distribuovaný systém přináší nové klíčové problémy, které by DDBMS měl umět řešit:

1. Distribuované SQL
2. Distribuované transakce
3. Podpora distribuování dat
   - Fragmentace dat
   - Replikace dat
4. Zotavení z nových druhů chyb (výpadek sítě, výpadek uzlu, uváznutí distribuované tansakce, distribuovaný deadlock)

Distribuované SQL

- SQL dotazy do lokálních tabulek jsou zpracovány lokálním DBMS (stejně jako by databáze nebyla distribuovaná)
- SQL dotazy do vzdálených tabulek (nebo kombinace lokálních a vzdálených tabulek) musí být rozloženy na dílčí operace v jednotlivých databázích (podle umístění dat)
- Nemělo by být nutné přenášet všechna data do jednoho uzlu a tam zpracovávat dotaz
- Možný paralelismus

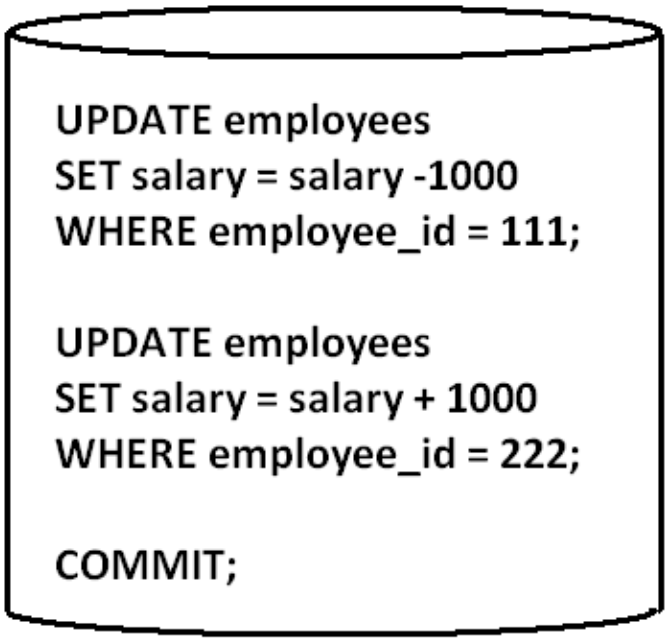
Optimalizace SQL

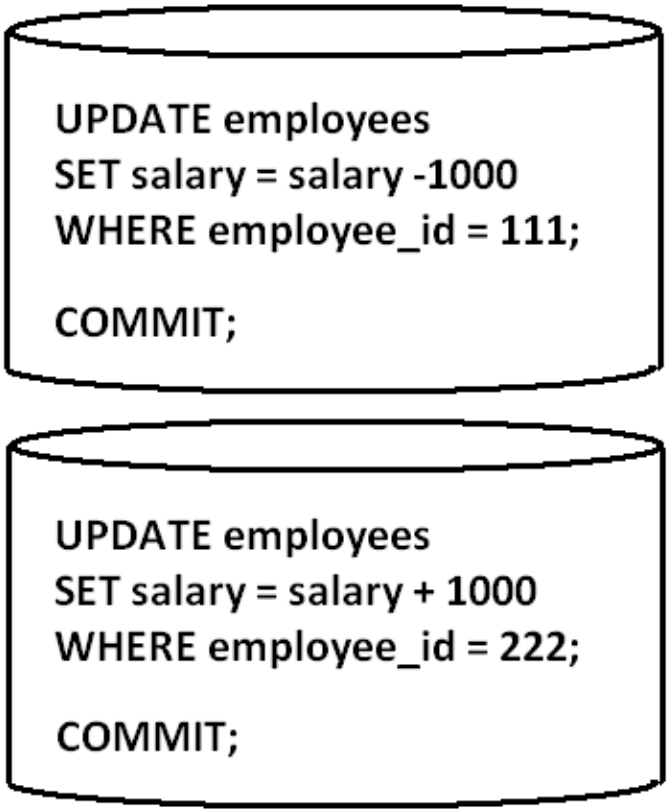
- Stanovení optimálního exekučního plánu pro vykonání SQL
- Lokální optimalizace – I/O cost + CPU cost
- DDBMS - I/O cost + CPU cost + communication cost

```
SELECT EMPLOYEES.*,DEPARTMENTS.DEP_NAME
FROM EMPLOYEES JOIN DEPARTMENTS
ON EMPLOYEES.DEP_ID = DEPARTMENTS.DEP_ID
WHERE EMPLOYEES.LAST_NAME LIKE 'B%';
```

Distribuované transakce

- Musí být možné provést transakčně změnu dat ve více databázích najednou





2PC (Two Phase Commit)

- Protokol pro řešení distribuovaných transakcí
- Někdo to musí řídit – uzel koordinátor
- Ostatní uzly musí poslouchat
- Vlastní commit probíhá ve dvou fázích
- **Fáze PREPARE**
  - Koordinátor zašle zprávu PREPARE všem uzlům, které se účastní distribuované transakce (tj. kde byla provedena změna dat)
  - Sám koordinátor ve fázi PREPARE nic nedělá, jen čeká na výsledky od ostatních uzlů

2PC (Two Phase Commit)

- Každý uzel se pokusí provést fázi PREPARE – uloží transakční log na disk, neprovádí ale vlastní commit, transakce je tedy stále neukončená, změněné řádky zůstávají stále zamčené, jen se zajistí, aby aktuální stav rozpracované transakce byl schopný přežít výpadek
- Každý uzel odpoví zpět PREPARED (pokud se vše povedlo) nebo ABORT (pokud došlo k problému)

- **Fáze COMMIT**
  - Koordinátor po obdržení vyhodnotí výsledky
  - Pokud jsou všechny výsledky PREPARED, provede potvrzení svých lokálních změn (standardní COMMIT) a všem uzlům zašle zprávu COMMIT

2PC (Two Phase Commit)

  - Pokud je alespoň jeden výsledek ABORT, provede ROLLBACK svých lokálních změn (standardní ROLLBACK) a všem uzlům zašle zprávu ROLLBACK
- Protokol je imunní proti výpadku sítě/uzlu v libovolné fázi
- Koordinátor si po celou dobu zpracování globální transakce udržuje data o stavu transakce a všech podtransakcích a je tedy schopen transakci dokončit např. po obnovení spojení se vzdáleným uzlem
- IN-DOUBT transakce (výpadek po PREPARED)

Distribuování dat

- Data je nutno rozmístit do jednotlivých databází tak, aby se minimalizovaly síťové přenosy a maximalizovalo lokální zpracování
- Per site constraints
- Uplatnění hlavně při návrhu shora-dolů
- Pro správný návrh je nutná analýza používání dat v jednotlivých uzlech
- Základní jednotka alokace je tabulka
- Další techniky pro alokaci dat jsou fragmentace a replikace

Fragmentace

- Vhodné, pokud se v každém uzlu distribuovaného systému pracuje s částí tabulky
- Rozdělení relace na několik sub-relací, které se umístí do jednotlivých databází
- Data se umístí tam, kde se nejčastěji používají, ostatní data jsou dostupná ve vzdálených databázích – omezení síťových přenosů
- Zvyšuje bezpečnost dat – data se neukládají tam, kde nejsou potřeba
- Musí být úplná, disjunktní, rekonstruovatelná

Fragmentace

- Horizontální –podmožina řádků

- Vertikální – podmožina sloupců (projekce)

Fragmentace

- Smíšená – horizontálně fragmentovaný vertikální fragment nebo vertikálně fragmentovaný horizontální fragment
- Odvozená horizontální – podřízená tabulka fragmentovaná podle horizontální fragmentace rodičovské tabulky

Replikace

- Udržování kopií tabulek nebo fragmentů ve více databázích
- Omezení síťových přenosů
- Zvýšení dostupnosti dat při výpadku části distribuovaného systému
- Náročná údržba kopií
- Vhodné pro často používaná data, která se málo aktualizují

Replikace

Podle těsnosti spojení replik

- Synchronní - aktualizace replik je součást transakce, degraduje výkon, spolehlivost
- Asynchronní - na vyžádání nebo v pravidelných intervalech, je nutno počítat s tím, že data nebudou vždy úplně aktuální

Podle způsobu provádění repliky

- Pouze změněné řádky – nutný log změněných řádků
- DML příkazy – fronta odložených příkazů
- Úplná

Replikace

Podle způsobu zacházení s replikami

- Pouze pro čtení (master-slave), změny je možné provádět jen v master, repliky jsou read-only
- Pro transakční zpracování (multi-master) – změny je možno provádět i v replikách, nutno synchronizovat obousměrně – mohou vznikat konflikty (změněn stejný řádek ve dvou různých replikách), řešení konfliktů - priority, časová razítka, manuální, vlastní procedura

12 pravidel DDBMS

Christopher Date's 12 rules:

1. Lokální autonomie – jednotlivé databáze by měly být maximálně samostatné (lokální data jsou řízena lokálně, lokální operace zpracovává lokální DBMS – žádná závislost na jiném uzlu)
2. Neexistuje žádné centrální místo, na kterém by byl distribuovaný systém závislý (tj. není možné, aby bylo například zpracování SQL dotazů prováděno nějakým centrálním uzlem)

12 pravidel DDBMS

3. Nepřetržitá činnost – neměl by existovat důvod k odstávce celého systému (např. při upgrade DBMS některé z databází)
4. Nezávislost na umístění dat – uživatel nemusí vědět, kde jsou data umístěna (umožňuje data přesouvat)
5. Nezávislost na fragmentaci dat
6. Nezávislost replikaci dat
7. Distribuované zpracování SQL
8. Distribuované zpracování transakcí

12 pravidel DDBMS

9. Nezávislost na HW
10. Nezávislost na OS
11. Nezávislost na síti
12. Nezávislost na konkrétním DBMS (nemusí jít ani o RDBMS)

0. DDBMS by se měl z uživatelského pohledu tvářit jako jako lokální DBMS

Jak to dělá Oracle

Oracle

- První distribuované vlastnosti koncem 80. let
- Skutečný DDBMS od verze 7 (1992)
- Využívá koncept, kdy databázový server zprostředkovává pro své klienty přístup na data jiných databázových serverů
- Databázové severy je nutné propojit – používá se stejný protokol jako pro komunikaci mezi klientem a serverem (Net8)
- Základ konceptu je databázový link

DB Link

- Databázový objekt (podobně jako třeba VIEW, ROLE…)
- Definuje připojení do vzdálené databáze
- Obsahuje identifikaci vzdálené databáze a databázový účet, na který se připojí
- DB Link se zakládá v databázovém serveru, který má zprostředkovat svým klientům přístup do vzdálené databáze (pro symetrické propojení nutné dva linky)
- Klient-server architektura (lokální databázový server se připojuje ke vzdálenému stejně, jako by to byl „obyčejný“ klient

DB Link

CREATE PUBLIC DATABASE LINK sample.cca.cz
USING ‘sample.cca.cz';

CREATE PUBLIC DATABASE LINK sample.cca.cz
CONNECT TO hr IDENTIFIED BY hr
USING ‘sample.cca.cz';

SELECT * FROM employees@sample.cca.cz;

INSERT INTO employees@sample.cca.cz (...)
VALUES (...);

DB Link

- Přes link lze:
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
  - COMMIT / ROLLBACK
- Přes link nelze:
  - DDL příkazy (CREATE TABLE …)
  - Přidělovat oprávnění (GRANT …)
  - Refereční integrita (FOREIGN KEY …REFERENCES …)

Location transparency

```
CREATE VIEW sample_employees AS
SELECT *
FROM employees@sample.cca.cz;
```

nebo

```
CREATE SYNONYM sample_employees
FOR employees@sample.cca.cz;
```

```
SELECT *
FROM sample_employees;
```

PL/SQL - RPC

```
BEGIN
   delete_employee@sample.cca.cz(1000);
END;

CREATE SYNONYM delete_employee
FOR delete_employee@sample.cca.cz;

CREATE PROCEDURE delete_employee(id NUMBER) IS
BEGIN
   delete_employee@sample.cca.cz(id);
END;
```

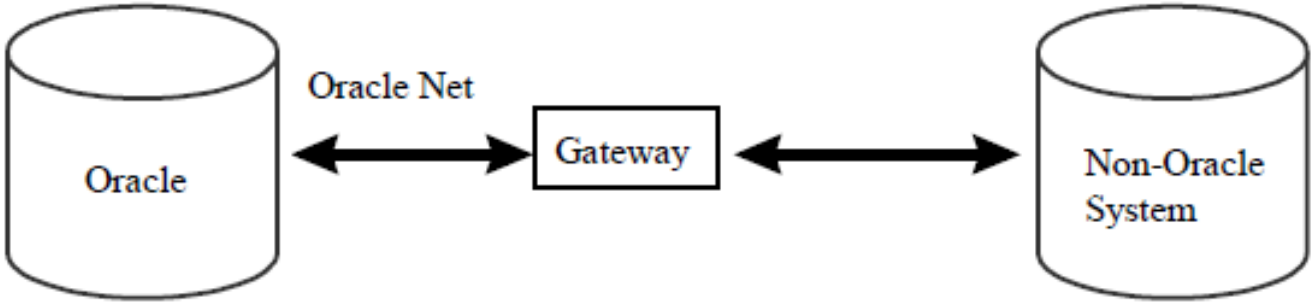
Bezpečnost

- Zpřístupnění vzdálených dat řídí administrátor vzdálené databáze – zakládá databázové konto, na které vede databázový link

  Přes databázový link je možno dělat jen to, co administrátor vzdálené databáze dovolí – je to standardní mechanismus přidělování práv jako pro každého jiného uživatele
- Protokol Net8 umožňuje šifrování – tj. data přenášená mezi databázemi v distribuované prostředí je možno zabezpečit šifrováním (v LAN nemusí být potřeba)

Oracle Gateways

- Pro heterogenní prostředí (různé DBMS)
- SW který zajistí, že se non-Oracle DBMS tváří jako Oracle



- Překlad SQL, datových typů, systémového katalogu, Distribuované transakce, RPC, optimalizace SQL
- DB2, Sybase, Informix, SQL Server, VSAM, Adabas, …
- Gateway for ODBC

Distribuované SQL

- Remote SQL dotaz – zpracování proběhne lokálně ve vzdálené databázi, zpět se předá výsledek

```
SELECT * FROM employees@sample.cca.cz;
```

- Distribuovaný SQL dotaz

```
SELECT emp.employee_name, dep.department_name
FROM employees@sample.cca.cz emp,
        departments dep
WHERE emp.department_id = dep.department_id;
```

- Naprosto transparentní, vše řeší DBMS

Distribuované transakce

- **Remote transakce** – lokální transakce ve vzdálené databázi

```
UPDATE employees@sample.cca.cz
SET salary = salary - 1000
WHERE employee_id = 111;

UPDATE employees@sample.cca.cz
SET salary = salary + 1000
WHERE employee_id = 222;

COMMIT;
```

Distribuované transakce

- **Distribuovaná transakce**

```
UPDATE employees@sample.cca.cz
SET salary = salary - 1000
WHERE employee_id = 111;

UPDATE employees
SET salary = salary + 1000
WHERE employee_id = 222;

COMMIT;
```

Distribuované transakce

- Naprosto transparentní, zápis v SQL je shodný jako pro lokální transakce
- Oracle sám rozhodne o použití 2PC, pokud transakce obsahuje změny ve více než jedné databázi
- Commit Point – omezení negativního vlivu IN-DOUBT transakcí. Vychází z toho, že koordinátor nemůže skončit ve stavu IN-DOUBT. Jako koordinátora distribuované transakce je tedy vhodné vybrat databázi s nejdůležitějšími daty (ve smyslu konkurečního přístupu k datům). Výběr koordinátora lze ovlinit nastavením priority do inicializačního parametru databáze COMMIT_POINT_STRENGTH.

Distribuované transakce

- DBA_ 2PC_PENDING, DBA_2PC_NEIGHBORS – informace o nedokončených distribuovaných transakcích
- RECO – proces který se stará o automatické dokončení distribuovaných transakcí po výpadku – spouští se v pravidelných intervalech a po obnovení spojení se vzdáleným systémem se pokusí o dokončení distribuované transakce
- Manuální dokončení IN-DOUBT transakce

COMMIT FORCE '*transaction_id*';

ROLLBACK FORCE '*transaction_id*';

Fragmentace

- Žádná speciální podpora
- Lze vyřešit pomocí standardních relačních operací JOIN, UNION + databázový link
- Transparentnost lze zajistit pomocí konceptu VIEW

```
CREATE VIEW all_employees AS
SELECT * FROM employees@sample.cca.cz
UNION ALL
SELECT * FROM employees;
```

Replikace

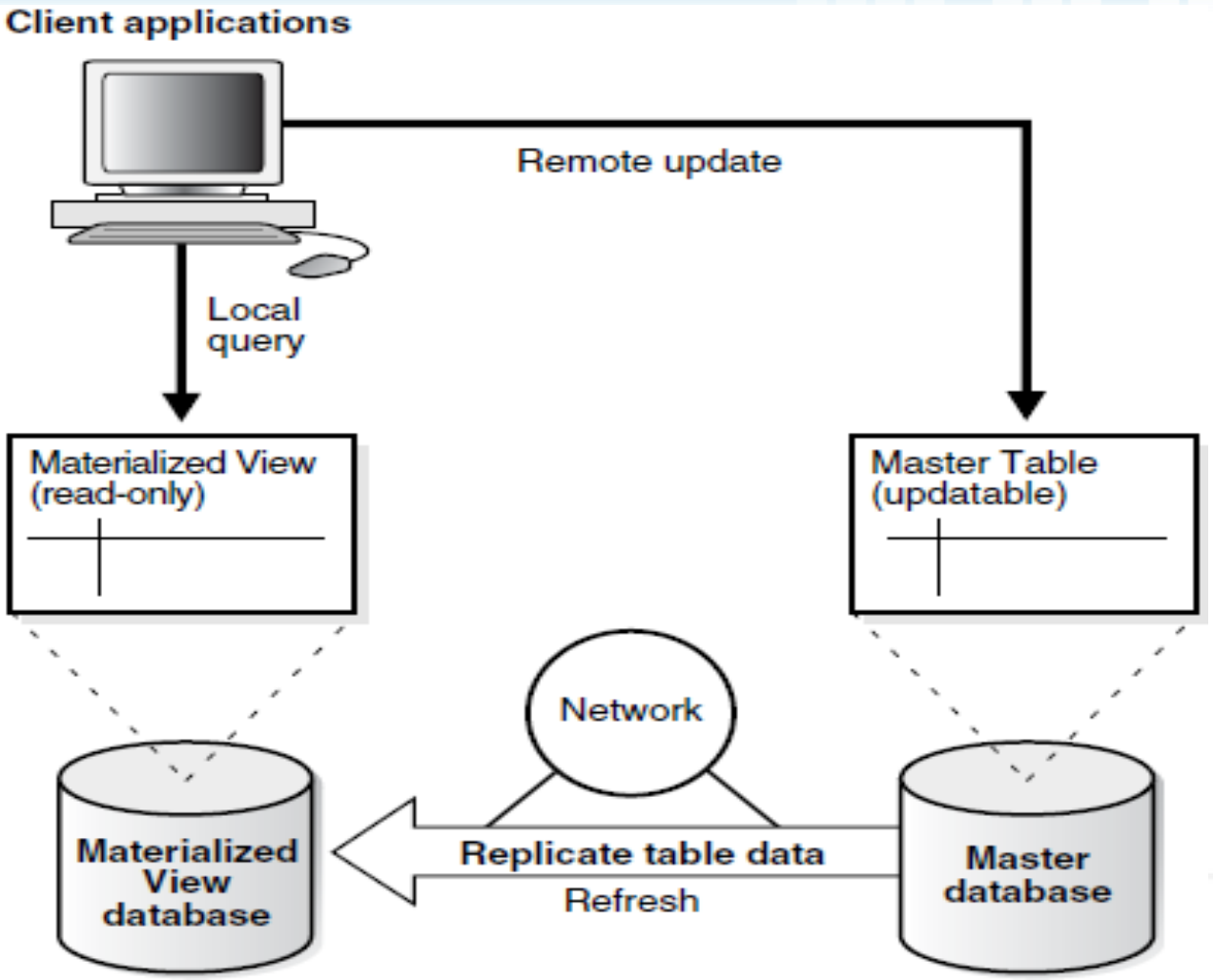
- Master-Slave
  - Read-only materialized view
  - Updateable materialized view
- Multi-master replication

MATERIALIZED VIEW

- Koncept podobný jako VIEW – tj. uložený dotaz, se kterým lze pracovat jako s tabulkou
- Data však nejsou dotazována online jako při použití VIEW, ale jsou uložena v pomocné tabulce
- Refresh těchto dat lze provádět automaticky se zadaným intervalem nebo na vyžádání
- Pro uživatele naprosto transparentní – vůbec nepozná, že nepracuje s tabulkou, ale se snapshotem
- Data jsou read-only
- Možno použít také jako cache pro náročné dotazy (data warehouse)
- Nevyžadují stále propojení databází – jen pro refresh

CCA GROUP

MATERIALIZED VIEW



MATERIALIZED VIEW

```
CREATE MATERIALIZED VIEW employees
REFRESH FAST
START WITH SYSDATE
NEXT SYSDATE+1/24
AS  SELECT * FROM employees@sample.cca.cz;
```

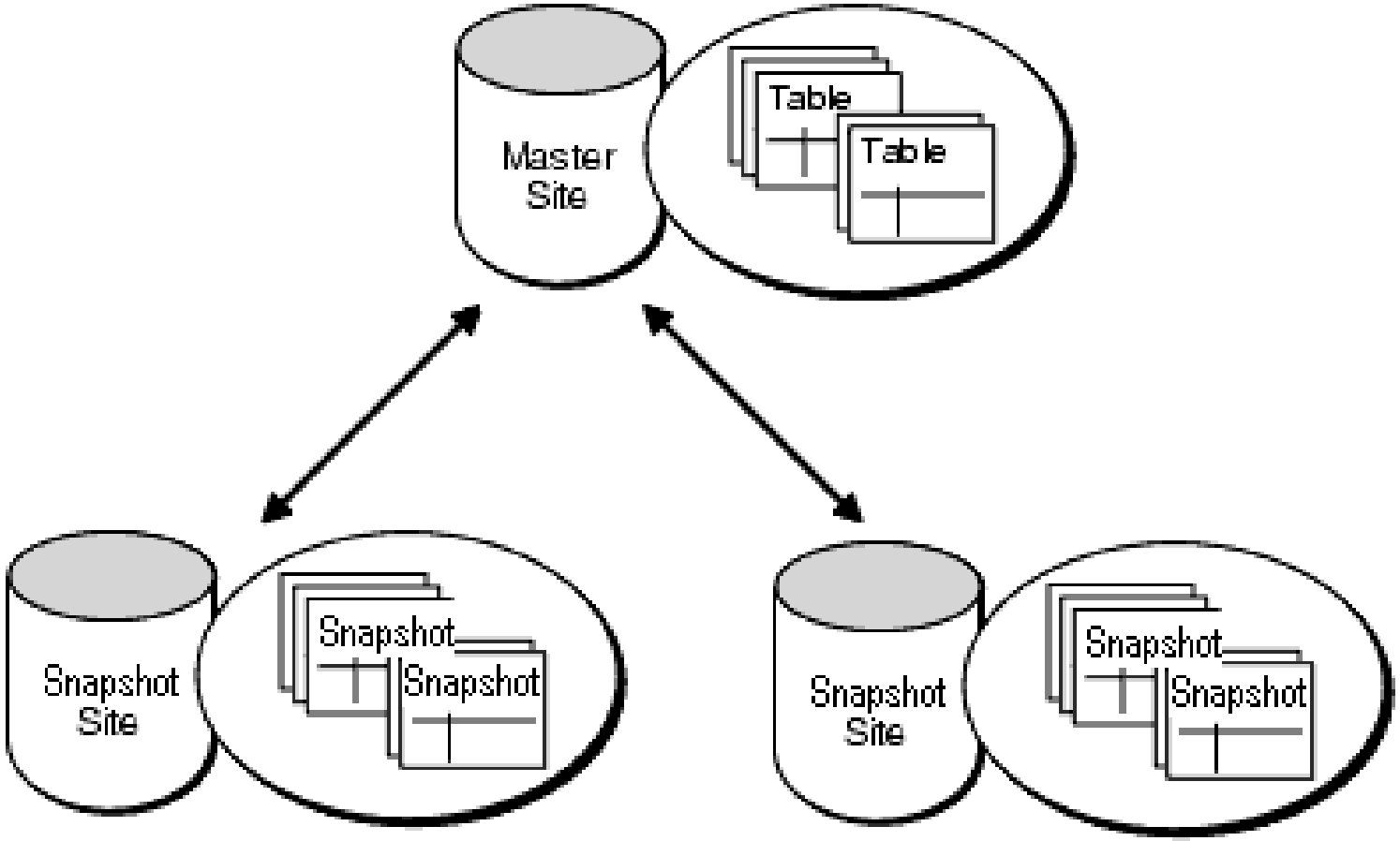
- Refresh
  - Fast (nutný log změných řádků – PK + I, U, D)
  - Complete
- DBMS_MVIEW.REFRESH(‘employees’) – refresh na žádost
- Refresh group – skupina materializovaných view, které se refreshují najednou (data k jednomu okamžiku)

Updateable MVIEW

```
CREATE MATERIALIZED VIEW employees
FOR UPDATE
AS SELECT * FROM employees@sample.cca.cz;
```

- Vyžaduje Oracle Advanced Replication
- Umožňuje měnit data (insert, update, delete) snapshotu a při replikaci se tyto změny promítnou do master tabulky
- Mohou vznikat kolize – více změn jednoho záznamu
- Je možné mít celou řadu databázových uzlů se snapshotem stejné master tabulky, ty offline updatovat (mobilní zařízení) a pak po připojení k síti provést refresh

Updateable MVIEW



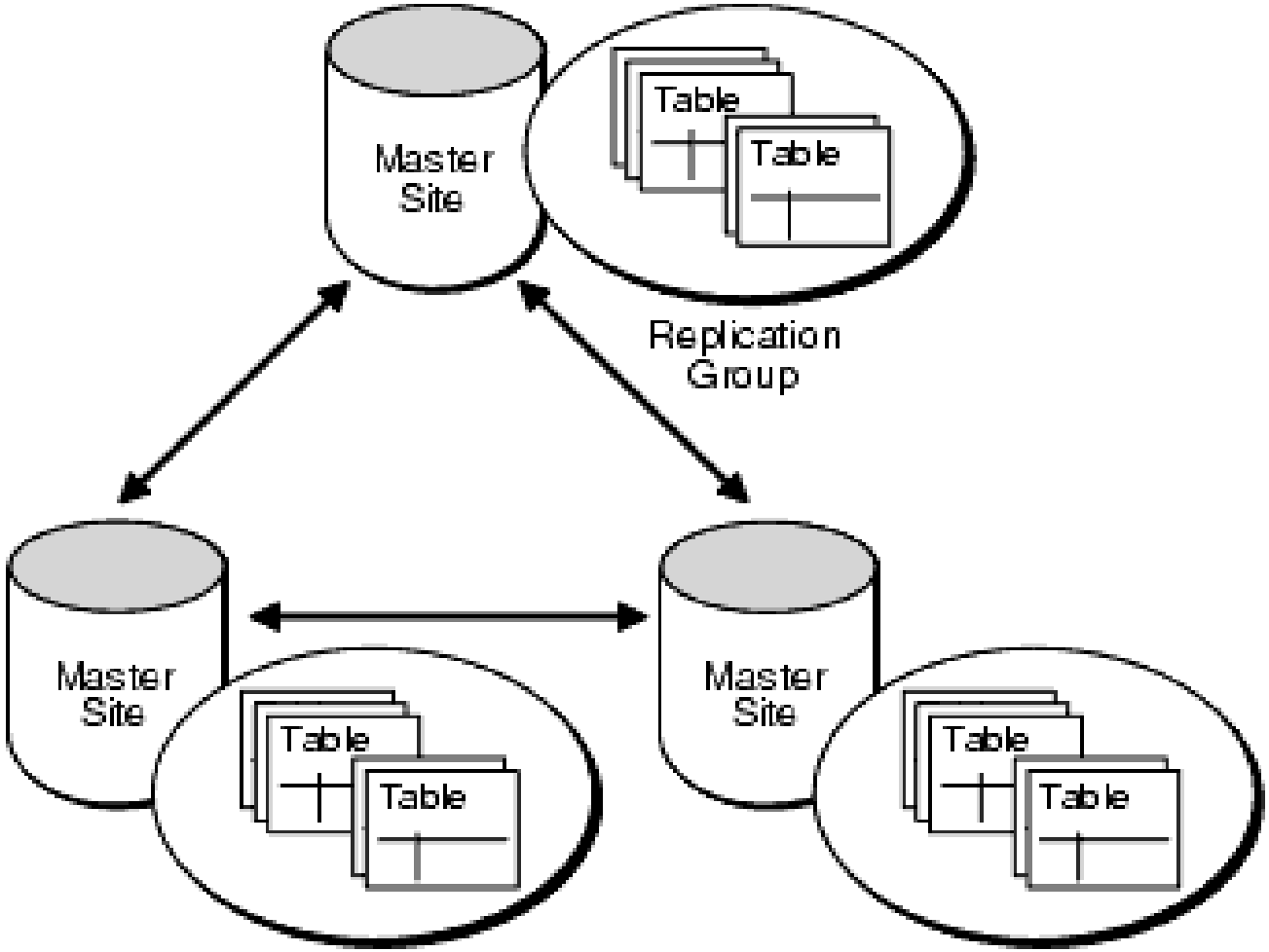
Řešení kolizí

- Update kolize - více změn jednoho záznamu v časovém rozmezí jednoho refresh

  (Timestamp, Site priority, Overwrite, Average, Maximum, Minimum)
- Kolize unikátnosti – vložení záznamu se stejným PK

  (Discard, Append Sequence, Append Site ID)
- Delete kolize – smazání záznamu, který je v jiném uzlu později změněn
- Je možné psát vlastní obsluhu kolizí

Multi-master replikace



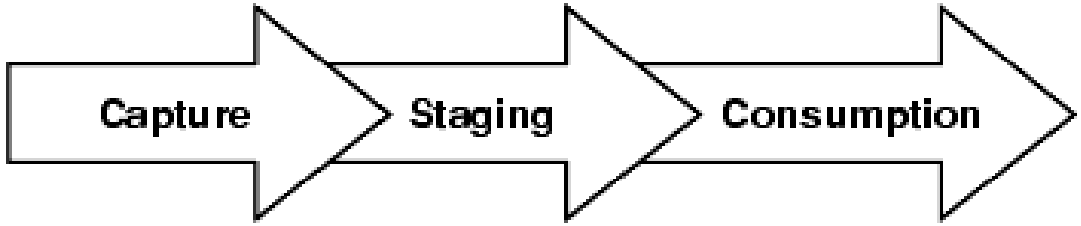
Multi-master replikace

- Několik rovnocenných replik jedné tabulky
- Všechny repliky mohou být updatovatelné
- Změny se automaticky promítají do všech replik
- Kolize se řeší stejně jako u updateable MVIEW
- Asynchronní propagace změn
  - Změny v tabulce se ukládají do fronty (odložené DML)
  - Při refresh (interval, ruční vyvolání) se změny promítnou do replik
- Synchronní propagace změn – vynechá se fronta, údržba replik je ve stejné transakci jako původní DML příkaz. Nutná trvalá dostupnost všech uzlů !

CCA GROUP

Další způsoby replikace

- Oracle Streams – obecný mechanismus pro toky informací (ne jen pro replikace)



- Replikace vlastními prostředky
  - Synchronní – row level trigger + distribuovaná transakce
  - Asynchronní – row level trigger + fronta + naplánovaná úloha

DDBMS 12 rules

1. Lokání autonomie – pokud není v příkazu použit databázový link (@), je zpracování výhradně lokální, správa dat a přístupů zůstává lokální
2. Nezávislost na centrálním uzlu
3. Nepřetržitá činnost – neexistuje žádný důvod k odstávce celého systému
4. Nezávislost na umístění dat – uživatel se nemusí starat o to, kde jsou data fyzicky uložena. Používá symbolické pojmenování (link). Navíc je možno mu tuto informaci úplně skrýt (synonymum, view).
5. Nezávislost na fragmentaci dat – koncept VIEW

Oracle DDBMS 12 rules

6. Nezávislost na replikaci dat - s replikou se vždy pracuje jako s normální tabulkou
7. Distribuované zpracování SQL – naprosto transparentní, uživatel nemusí nijak řešit, jen použije v SQL příkazu link
8. Distribuované zpracování transakcí - 2PC, naprosto transparentní, uživatel používá COMMIT tak jak je zvyklý
9. Nezávislost na HW
10. Nezávislost na OS
11. Nezávislost na síti (různé sítě řeší Net8)
12. Nezávislost na konkrétním DBMS

Dotazy ?