Databázové systémy

09.indexy, transakcie, úvod do bezpečnosti DBS

KKUI
Katedra kybernetiky
a umelej inteligencie

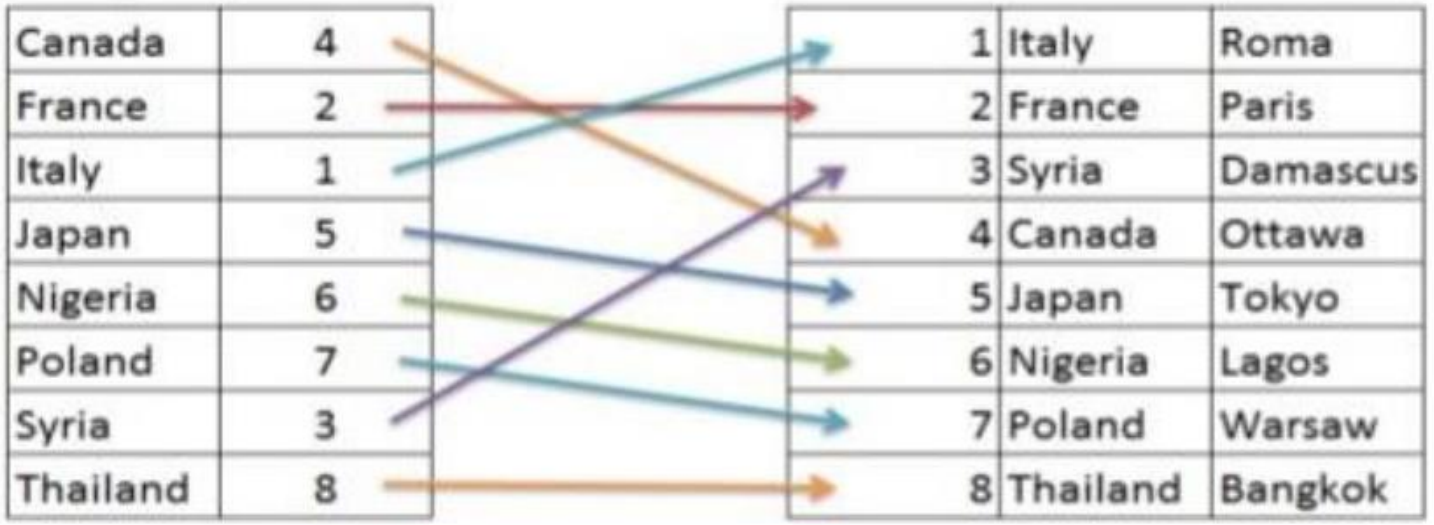
Indexy

- databázový objekt používaný na urýchlenie vyhľadávania v tabuľkách
- zaraďuje kľúčový atribút s jeho hodnotou ROWID, ktorá prepojí index so záznamom v tabuľke
- nad jednou tabuľkou môže existovať viacero indexov vytvorených nad jedným alebo viacerými stĺpcami tabuľky
- každý stĺpec môže byť indexovaný iba raz
- o použití indexu rozhoduje optimizer
- použitie indexu môže byť vynútené pomocou HINTU
- spomalenie pri úprave primárnych dát
- môže byť väčší ako tabuľka nad ktorou je vytvorený

Indexy

What is an Index

- Data structure to improve the speed of data retrieval from DBs.



Canada	4
France	2
Italy	1
Japan	5
Nigeria	6
Poland	7
Syria	3
Thailand	8

1	Italy	Roma
2	France	Paris
3	Syria	Damascus
4	Canada	Ottawa
5	Japan	Tokyo
6	Nigeria	Lagos
7	Poland	Warsaw
8	Thailand	Bangkok

Indexy – práca s indexmi

- vytvorenie/zmazanie indexu nemá vplyv na dáta iba na výkon
- najefektívnejšie je vytvárať indexy nad stĺpcami, ktoré sa používajú vo WHERE klauzule

```
CREATE INDEX hr.dept_location_ix ON
  hr.departments (location_id);

DROP INDEX  hr.dept_location_ix;
```

Indexy – typy indexov

- algoritmy indexovania
  - Indexy B-stromu: predvolené a najbežnejšie
  - Bitmapové indexy: kompaktné; najlepšie pracovať pre stĺpce s malou sadu hodnôt
- jednoduchý index - vytvorený nad jedným stĺpcom, najjednoduchší a najčastejšie používaný

```
CREATE INDEX name ON table (column) ;
```

- zložený index - vytvorený z dvoch alebo viacerých stĺpcov tabuľky, poradie stĺpcov má vplyv na výkon

```
CREATE INDEX name ON table (column1, ...,
  columnN) ;
```

Indexy – typy indexov

- unikátne indexy - používaný kvôli výkonu a integrite údajov, nedovolí vytváranie duplikátov v tabuľke

```
CREATE UNIQUE INDEX name ON table (column);
```

- implicitné indexy - vytvárané automaticky db serverom pri vytvorení objektu napr.: primárny kľúč, unikátny kľúč
- indexy založené na funkciách: obsahujú predkompilovanú hodnotu funkcie / výrazu

```
CREATE INDEX first_name_idx ON user_data
(UPPER(first_name));
```

- globálne a lokálne indexy: vzťahujú sa na particiované tabuľky (globálny nad celou tabuľkou, lokálny nad konkrétnou partíciou)

Indexy

- pri vytvaraní indexu treba brať do úvahy:

- veľkosť tabuľky - čím menšia, tým menší efekt
- distribúcia údajov - index pomáha nájsť špecifický údaj
- záťaž v podobe dopytov vs. modifikácie - dopyty sú rýchlejšie, modifikácie spomalené
- funkčnosť a výkon indexu otestovať voči konkrétnym dopytom
- nad stĺpcami v primárnom kľúči sa index vytvára automaticky

Indexy

Aspekty v prospech indexu:

- cudzie kľúče (väčšina stĺpcov pre spojenie)
- stĺpce uvedené v ORDER BY alebo GROUP BY
- stĺpce obsahujúce jedinečné údaje
- stĺpce vyskytujúce sa za WHERE, ktoré vracajú malé množstvo riadkov

! vždy otestovať

Indexy

Aspekty v neprospech indexu:

- malé tabuľky
- tabuľky, ktoré sú často aktualizované (riešenie: zmazanie a opätovné vytvorenie indexu
- stĺpce s veľkým obsahom NULL hodnôt
- stĺpce, s ktorými sa často manipuluje
- stĺpce, ktoré vracajú obrovské množstvo riadkov pri filtrovaní výsledku, napr. pohlavie

Execution plan

```
SQL>  explain plan set statement_id='st1' for select * from employees where department_id=20;

Explained.
SQL> SELECT PLAN_TABLE_OUTPUT
  FROM TABLE(DBMS_XPLAN.DISPLAY());  2

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------
Plan hash value: 1445457117

-------------------------------------------------------------------------------------------
| Id  | Operation                  | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |           |     2 |   266 |     3(0)   | 00:00:01 |
|*  1 |  TABLE ACCESS FULL         | EMPLOYEES |     2 |   266 |     3(0)   | 00:00:01 |
-------------------------------------------------------------------------------------------

SQL> create index emp_dep on employees(department_id);

Index created.
SQL>  explain plan set statement_id='st1' for select * from employees where department_id=20;

Explained.

SQL> SELECT PLAN_TABLE_OUTPUT
  FROM TABLE(DBMS_XPLAN.DISPLAY());  2

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------
Plan hash value: 3518676727

-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name      | Rows | Bytes | Cost (%CPU)| Time    |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |           |     2|   266 |     2   (0)| 00:00:01|
|   1 |  TABLE ACCESS BY INDEX ROWID  | EMPLOYEES |     2 |   266 |     2   (0)| 00:00:01|
|*  2 |   INDEX RANGE SCAN            | EMP_DEP   |     2 |       |     1   (0)| 00:00:01|
-------------------------------------------------------------------------------------------
```

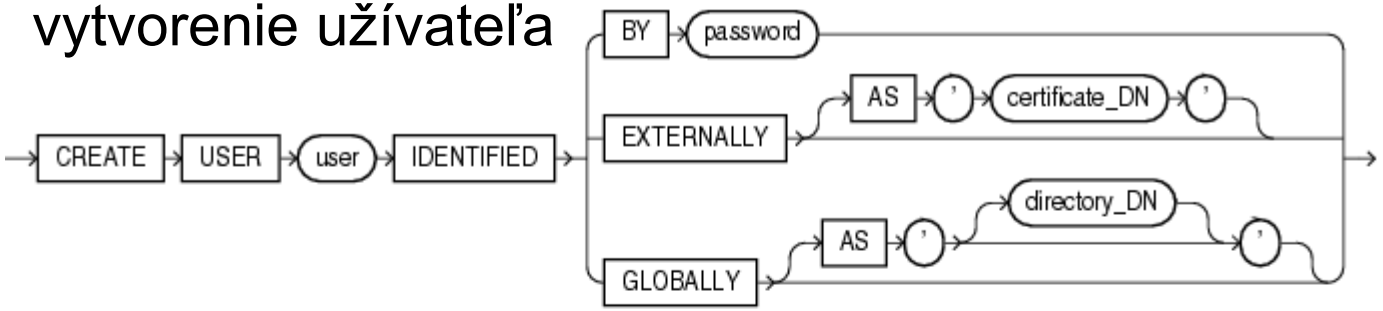
Zabezpečenie DBS

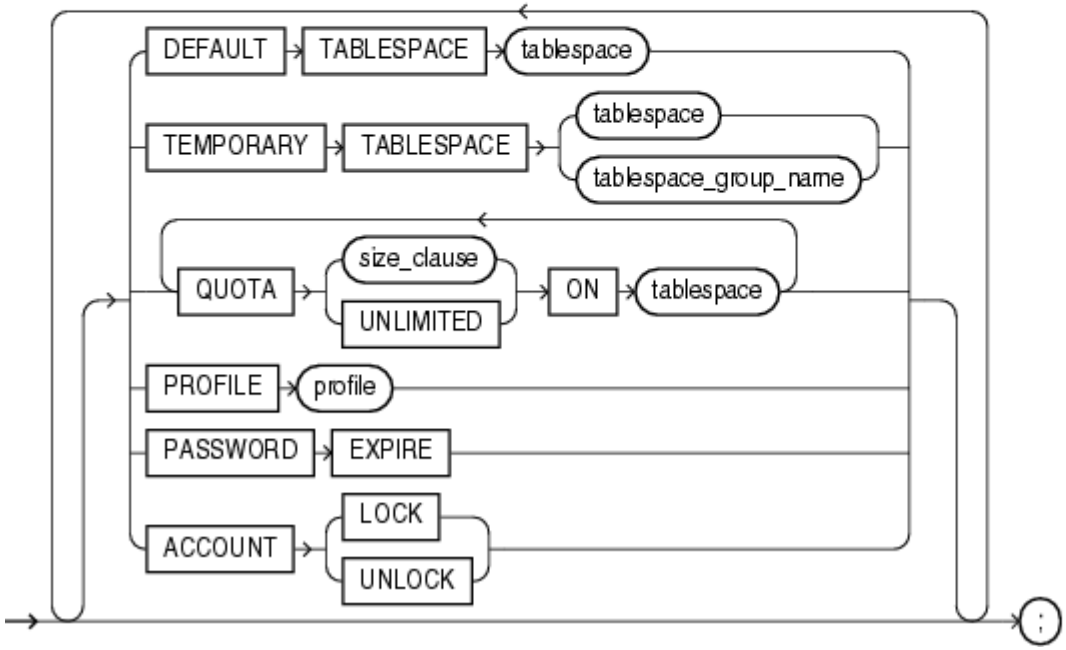
- zabezpečenie=

  bezpečnosť systému(DBA) + bezpečnosť dát
- bezpečnosť dát úzko súvisí so správou užívateľov
- nástroje a mechanizmy zabezpečenia dát (pred poškodením, zneužitím, ..):
  - ***autentifikácia:*** zahŕňa všetky metódy na overenie identity užívateľov
  - ***autorizácia:*** umožňuje prístup k objektom DB
  - ***audit:*** sledovanie DB ako úspešné/neúspešné prihlásenie, štatistika prístupu k objektom, realizácia konkrétnych akcií, ...

Správa používateľov

- vytvorenie užívateľa





Správa používateľov

Príklad:

```
CREATE USER janko IDENTIFIED BY hrasko321;
CREATE USER janko IDENTIFIED BY hrasko321
    DEFAULT TABLESPACE users;
CREATE USER janko IDENTIFIED BY hrasko321
    DEFAULT TABLESPACE users
    ACCOUNT LOCK PASSWORD EXPIRED;
```

Prihlásenie:

OS: sqlplus <username>/<password>

sqlplus: connect <username>/<password>

Správa používateľov

- zmazanie používateľa

  DROP USER <meno používateľa> [CASCADE];

  Príklad:

  ```
  DROP USER janko;
  DROP USER oe CASCADE;
  ```

- úprava používateľa

  ALTER USER <meno používateľa> [OPTIONS];

  Príklad:

  ```
  ALTER USER janko IDENTIFIED BY hrasko123;
  ALTER USER janko PASSWORD EXPIRE;
  ```

Privilégia

- riadia prístup k DB, objektom DB a na manipuláciu s dátami
- priradenie privilégia: GRANT, odobratie: REVOKE
- typy privilégií:
  - ***objektové:*** umožňujú užívateľom vykonávať operácie nad konkrétnymi objektmi, napr: select nad tabuľkou Employees
  - ***systémové:*** umožňujú užívateľom vykonávať operácie nad DB, napr.: prihlásenie do DB, vytvoriť/odstrániť DB, vytvoriť/odstrániť objekt, zmeniť objekt,...

Systémové privilégia

Syntax:

```
GRANT <privilege> TO <grantee>
   [WITH ADMIN OPTION];
REVOKE <privilege> FROM <grantee> ;
```

- príklady privilégií
  - CREATE SESSION
  - CREATE TABLE
  - ALTER ANY INDEX
  - DROP ANY SEQUENCE
  - EXECUTE ANY PROCEDURE
  - ALTER SESSION

Objektové privilégia

Syntax:

```
GRANT <privilege> ON <object name> TO <grantee>
   [WITH GRANT OPTION];
REVOKE <privilege> ON <object name> FROM
   <grantee> ;
```

- príklady privilégií:
  - SELECT
  - INSERT[column]
  - UPDATE[column]
  - REFERENCES[column]
  - DELETE
  - ALTER
  - DROP
  - EXECUTE

Quota

- slúži na umožnenie zápisu do daného tabuľkového priestoru(tablespace)

```
ALTER USER <user_name> QUOTA UNLIMITED
  ON <tablespace_name>;
ALTER USER <user_name> QUOTA 1M ON
  <tablespace_name>;
```

ROLE

- množina privilégií

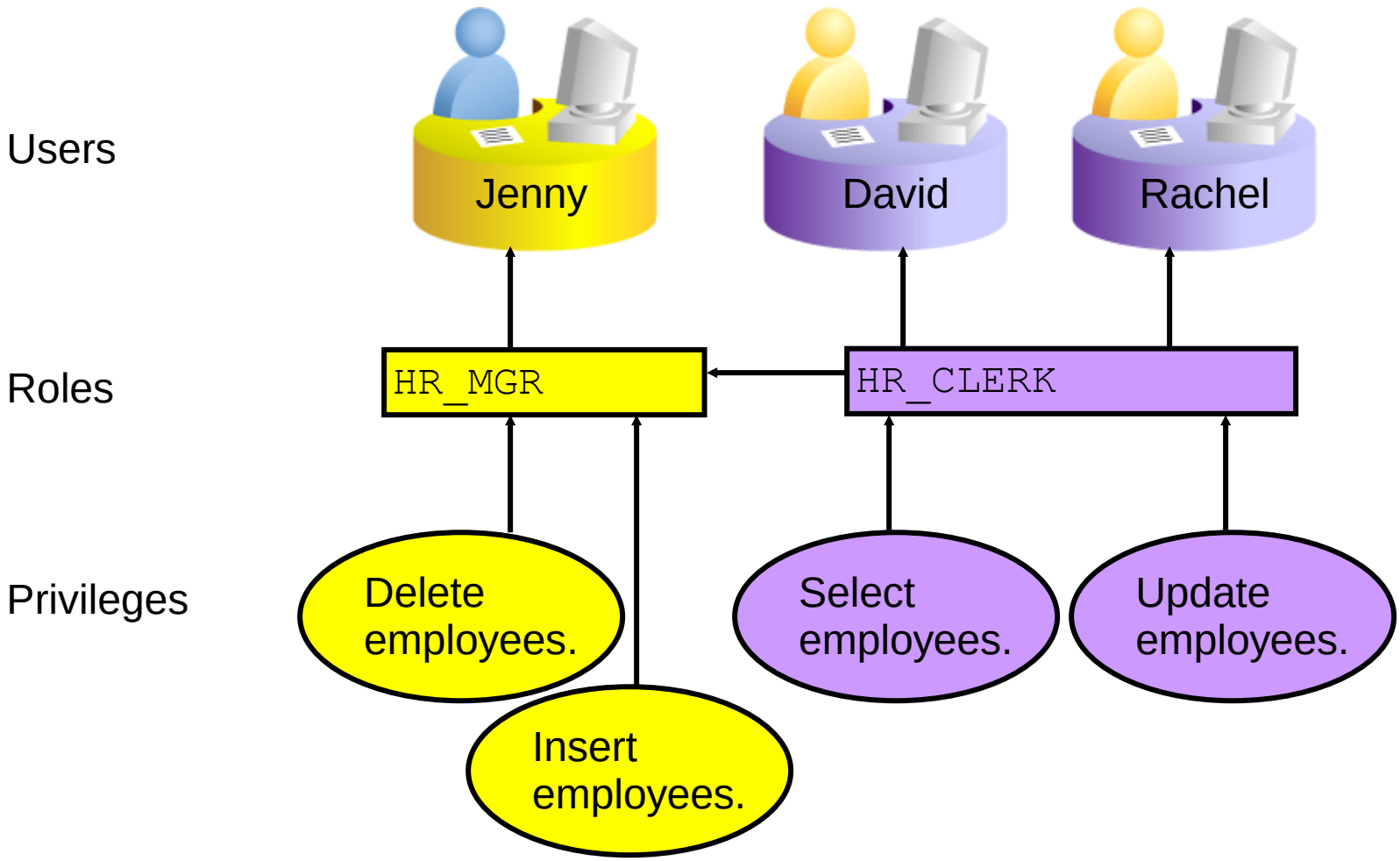
SYNTAX:

CREATE ROLE <role nam>;

Príklad:

```
CREATE ROLE create_object;
GRANT create table, create view, create
  sequence, alter any table TO create_object;
REVOKE alter any table FROM create_object;
```

ROLE


Users
Roles
Privileges
Jenny
David
Rachel
HR_MGR
HR_CLERK
Delete employees.
Insert employees.
Select employees.
Update employees.

Predefinované role

ROLE	GRANT/ROLE
CONNECT	CREATE SESSION
RESOURCE	CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
DBA	Most system privileges, several other roles. Do not grant to nonadministrators.
SELECT_CATALOG_ROLE	No system privileges, but HS_ADMIN_ROLE and over 1,700 object privileges on the data dictionary

Transakcie - definícia

- postupnosť jedného alebo viacerých SQL príkazov, ktoré sú prezentované ako logická a ucelená jednotka
- Ciele transakcií:
  - zdieľaný súčasný prístup na databázu
    - databázový systém má podporovať paralelný prístup viacerých používateľov, zároveň však musí dodržať integritu a konzitenciu údajov
  - odolnosť voči systémovým zlyhaniam
    - systém má zabezpečiť konzistenciu údajov aj pri systémovom zlyhaní

Transakcie

- každá transakcia má začiatok a koniec a je je pridelený jednoznačný identifikátor – transaction ID
- začiatok transakcie:
  - DDL príkaz
  - DML príkaz
  - príkaz SET TRANSACTION
- Koniec transakcie:
  - COMMIT/ROLLBACK
  - DDL príkaz
  - ukončenie session používateľom
  - neočakávané ukončenie session

Transakcie

- **Commit**
  - ukončí aktuálnu transakciu a vykoná všetky zmeny
  - všetky savepoint-y sú zmazané a transakčné zámky uvoľnené
- **Rollback (to Savepoint)**
  - odvolá všetky zmeny dát vykonané od začiatku transakcie resp. všetky zmeny až po SAVEPOINT určený jeho názvom
- **Savepoint**
  - označenie v transakcii, ku ktorému sa vieme vrátiť
- **Autocommit**
  - SET AUTOCOMMIT IMMEDIATE – zmeny sa vykonajú okamžite, každý príkaz je transakciou
  - SET AUTOCOMMIT OFF – zmeny sa ukladajú v tzv. UNDO segmentoch a v tabuľkách sa zapíšu po zadaní príkazu COMMIT alebo po korektnom ukončení práce v konzole SQLPLUS (exit)

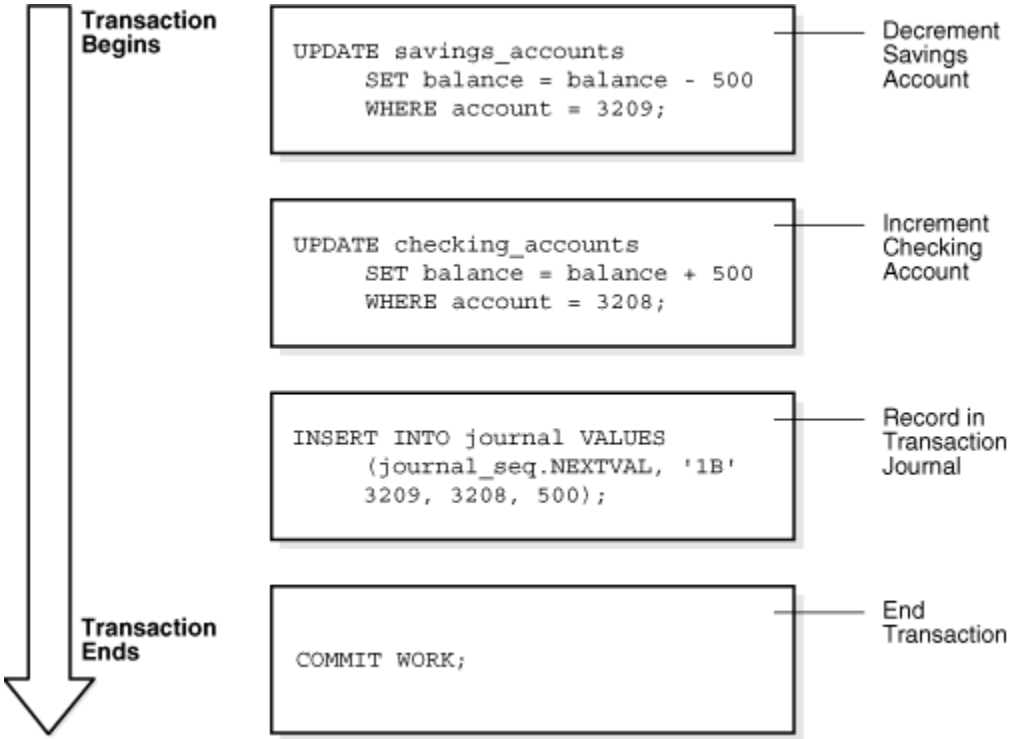
Transakcie – vlastnosti

- **A**tomicity
  - transakcia sa vykoná kompletne alebo vôbec
  - Example: Transaction update (100 rows) -> system fails after 20 updates -> DB rolls back changes.
- **C**onsistency
  - transakcia prevedie databázu z jedného konzistentného stavu do druhého t.j. pred jej začatím aj po ukončení sú dodržané všetky obmedzenia

Transakcie – vlastnosti

- **I**solation
  - výsledok transakcie nie je viditeľný kým ju používateľ nepotvrdí
  - dve transakcie sa nemôžu navzájom ovplyvniť
  - Example: Two users updating different files will not see uncommitted changes until done.
- **D**urability
  - po commite sa všetky zmeny stavajú trvalými
  - ukončenú transakciu už nie je možné prerušiť alebo zmeniť

Transakcie - príklad



Transakcie - príklad

T	Session	Explanation
t0	`COMMIT;`	This statement ends any existing transaction in the session.
t1	`SET TRANSACTION NAME 'sal_update';`	This statement begins a transaction and names it `sal_update`.

t2	`UPDATE employees`
	`  SET salary = 7000`
	`  WHERE last_name = 'Banda';`
		This statement updates the salary for Banda to 7000.

t3	`SAVEPOINT after_banda_sal;`	This statement creates a savepoint named `after_banda_sal`, enabling changes in this transaction to be rolled back to this point.

t4	`UPDATE employees`
	`  SET salary = 12000`
	`  WHERE last_name = 'Greene';`
		This statement updates the salary for Greene to 12000.

t5	`SAVEPOINT after_greene_sal;`	This statement creates a savepoint named `after_greene_sal`, enabling changes in this transaction to be rolled back to this point.

Transaction Control

Transakcie - príklad

t6	`ROLLBACK TO SAVEPOINT after_banda_sal;`	This statement rolls back the transaction to t3, undoing the update to Greene's salary at t4. The `sal_update` transaction has *not* ended.
t7	`UPDATE employees SET salary = 11000 WHERE last_name = 'Greene';`	This statement updates the salary for Greene to 11000 in transaction `sal_update`.
t8	`ROLLBACK;`	This statement rolls back all changes in transaction `sal_update`, ending the transaction.
t9	`SET TRANSACTION NAME 'sal_update2';`	This statement begins a new transaction in the session and names it `sal_update2`.
t10	`UPDATE employees SET salary = 7050 WHERE last_name = 'Banda';`	This statement updates the salary for Banda to 7050.
t11	`UPDATE employees SET salary = 10950 WHERE last_name = 'Greene';`	This statement updates the salary for Greene to 10950.
t12	`COMMIT;`	This statement commits all changes made in transaction `sal_update2`, ending the transaction. The commit guarantees that the

Transakcie - deadlock

Session 1	Session 2
SQL> UPDATE employees SET salary = salary*1.1 WHERE employee_id = 100 1 row updated.	SQL> UPDATE employees SET salary = salary*1. WHERE employee_id = 200 1 row updated.
SQL> UPDATE employees SET salary = salary*1.1 WHERE employee_id = 200 -- prompt does not return	SQL> UPDATE employees SET salary = salary*1.1 WHERE employee_id = 100 -- prompt does not return