10. Лекция

Category	Empty
Files	Empty
Created	May 28, 2023 11:22 AM
Reminder	Empty
Status	Open
URL	Empty
Updated	May 28, 2023 11:22 AM

Пада се - нормализация, релационен модел, релаицонна алгебра, ФЗ

- от колега, ходил на изпит 2 пъти

Транзакции

Въпрос:

Какви проблеми могат да възникнат?

Какво се случва, когато една реална система работи, какви проблеми могат да възникнат?

- аномалии при промяна на данните - ако изтрием нещо и с него махнем друго, което не сме искали да махаме
- failure -
- сигурност на данните
- противоречива информация - това е човешка грешка, неконсистентност на информацията

- проблеми при използването от много потребители
  - може да се претовари системата - невъзможност да обработи всички заявки, които получава
  - когато имаме едновременен достъп от много потребители, трябва да се направи някаква **синхронизация/ред**

Какво се случва, когато някои искат да четат(получават справки), а други да променят по някакъв начин БД

Когато имаме промени на данните, кога ги записваме - веднага или ги събираме и след време ги отразяваме на диска.

Има различни стратегии за решаването на този проблем

Overview of Transaction process

Ще разгледаме как oracle е решил проблемът

Когато работим с оператора SELECT, то само четем данните, но не ги и променяме

DML - data manipulation language

- insert, update, delete

Когато става дума за достъп до едни и същи данни, то трябва да предвидим различни възможни ситуации

concurrent DB access

```
update PIB _account Set money =
money - 500 where cID = 1235867
```

```
update PIB _account Set money =
money + 759 where cID = 1235867
```

Какво трябва да направим, за да отразим двете заявки, трябва да осъществим някакъв ред

През това време мениджърът на фирмата може да иска да провери този баланс - трети потребител, който иска достъп до тези данни

Трябва да направим механизъма достатъчно удобен и за работа и да не се губи консистентонстта на данните

Bulk Inserts

```
INSERT INTO Studio(name) SELECT DISTINCT studioName FROM Movie WHERE
studioName NOT IN (SELECT Name FROM Studio)
```

Какво ще стане, ако при въвеждането на данните стане грешка?

Ще бъдат ли запазени промените до момента?

Решение - групиране на database operations в transactions

Деф: Транзакция

Transaction is a sequence of one or more (DML) SQL operations treated as a unit

Последователност от DML оператори, които се възприемат като една **неделима единица**

Транзакцията е механизъм за работа със самите данни

Ако не успеем да извършим всичките тези опреции докрай, то използваме механизъм, за да се върнем до първоначалното състояние

ACID Transactions - характеристики на транзакциите

A DBMS is expected to support ACID transactions.

1. **Atomic** - either the whole transaction is run or none

   Всички оператори или се изпълняват, или не се изпълнява нито една

2. **Consistent** - database constraints are preserved/

   Оставяме БД в консистентно(съгласувано) състояние. Ако пращаме пари, трябва да имаме консистентност в числата на балансите на двата акаунта - нашият да се намали, а другият да се увеличи

3. **Isolated** - different transactions may not interact with each other

   Изолираност от това какво искат да направят другите потребители със същите тези данни

4. **Durable** - effects of a transaction are not lost in case of a system crash

   Трайност на данните, след като е приключила транзакцията, всички промени са направени и трайно записани в БД

Safety & security

safety - да пазим сами данните от сривове, неконсистентност, нарушение на цялстност/интегритет, правим го поради грешки и инциденти

security of data - всичките опити да се злоупотреби с данните, които се съхраняват и използват

безопасност и сигурност?

System log

За да може една БД да следи какво се случва с данните е необходимо да се правят някакви записи, които да отчитат промените, които се правят в тези БД.

В тези log-ове трябва да бъдат отбелязани всички възможни действия, които се извършват върху тези данни.

Атомарността изисква да можем да се върнем назад - затова и ни трябват какви са извършените промени

How is the log file used

- All permanent changes to data are recovered
    - possible to undo changes to data

Механизъмът за извършване на транзакцията изисква да можем да се върнем до състоянието преди началото на транзакцията

- необходимо е да пазим копие на данни, което е състоянието на данните преди да почнем тази транзакция
- в log файловете записваме всички действия, които сме предприели, за да променим данните
- за да може, ако се наруши целостта на операцията да се върнем назад и да възстановим информацията

Controlling transactions

Можем **експлицитно** да започнем транзакция, използвайки 'START TRANSACTION' израз

И **експлицитно** да я приключим с COMMIT или ROLLBACK

- **commit** causes an SQL transaction to complete successfully
    - any modifications done in the transaction become permanent in the database
- **rollback** causes an SQL transaction to end by aborting it
    - any modifications to the database must be undone
    - could be caused implicitly by errors

Oracle има "**Позитивна стратегия**" - счита, че при правене на промени, ние сме ги обмислии.

Може да считаме, че Oracle приема данните паралелно с всяка DML операция.

Ако обаче се откажем, той трябва да се върне и отново да запише данните в стария им вид

Microsoft прави обратното - не променя данните до въвеждането на оператора commit

Пази ги като копие и при commit записва данните в самата БД.

Oracle Database Transaction

Имплицитна транзакция

В момента, в който **oracle види DML операция**, той автоматично стартира транзакция.

Тя завършва явно при commit или rollback

Неявно завършване на транзакция

Неявно завършване на транзакция, когато види **първата DDL** (data definition language) или **DCL** (data control language) операция

DDL - създава нов обект в релационната схема на БД

- щом ще се променя релационната схема с БД, то той иска първо да приключи транзакциите, за да обърне цялото внимание, за промяната

DCL - създаваме потребители и им даваме права

Може да **завърши транзакция** и ако системата **crash**-не

Това е механизъм на защита, който oracle ползва.

Ако експлицитно започнем транзакция и срещне DDL, DCL операция, то то пак ще завърши неявно. Така го прави Oracle

В класическият случай ROLLBACK ни връща в началото на транзакцията, но има възможност да сложим междинни точки на съхранение (savepoint)

След това като решим да се върнем, да не се връщаме до самото начало, а само до определено място

Особено важно при масови наливания на данни, прехвърляния на данни, когато отнема много време

- понякога падат сървъри, спира ток, може да се случи инцидент

```
-- UPDATE .. SAVEPOINT update_done -- Insert .. ROLLBACK TO update_done
```

Implicit transaction processing

Ако нямаме commit или rollback, но СУБД-то срещне някоя от командите от DDL или DCL, то се прави неявен **commit**.

State of the Data before commit or rollback (Oracle)

Какво се случва с данните, които променяме по време на една транзакция в Oracle

- the previous state of the data can be recovered - предишното състояние може да бъде възстановено
- the current user can review the results of the DML operations by using the **select** statement
  - ние виждаме и редактираме оригиналните данни
- other users **cannot view** the results of the DML statements by the current user
  - Другите виждат направеното копие на данните
- the affected rows are **locked**; other users **cannot change data in the affected rows**

В моментът, в който приключи транзакцията, другите потребители вече виждат променените данни, могат да ги променят с техни си транзакции.

Ако решим да се върнем назад и да се откажем от тези промени, то oracle трябва да мине през log-файловете и да възстанови старото състояние на данните

Обратната стратегия - добра когато се връщаме назад, но ако имаме страшно дълги транзакции и напишем commit, започва едно интезивно записване в диска

State of the data after COMMIT

- data changes are made permanent in the database
- the previous state of the data is permanently lost
- all users can view the results
- locks on the affected rows are released; those rows are available for other readers to manipulate
- all savepoints are erased

Примери

```
DELETE FROM copy_emp -- forgets where and deletes all rows accidentally
ROLLBACK; -- restores the deleted data rows
```

```
DELETE FROM test; ROLLBACK; DELETE FROM test WHERE id = 100; SELECT * FROM
test WHERE id = 100; COMMIT;
```

Statement-level rollback

Около всяко DML statement се прави неявна точка на запазване

- if a single DML statement fails during execution, only that statement is rolled back

Това може и да се е променило, ще се придържаме към общите правила

Read consistency

- read consistency guarantees a consistent view of the data at all times
- changes made by one user do not conflict with changes made by another user
- read consistency ensures that on the same data:
  - readers do not wait for writers - четат копието, запазено преди началото на транзакцията
  - writers do not wait for readers - не чакат четящите, като искат да четат, да четат старото състояние, а в момента, в който си свършим работата ще видят новото състояние

Implementation of read consistency

Промените се нанасят в самите данни - data blocks

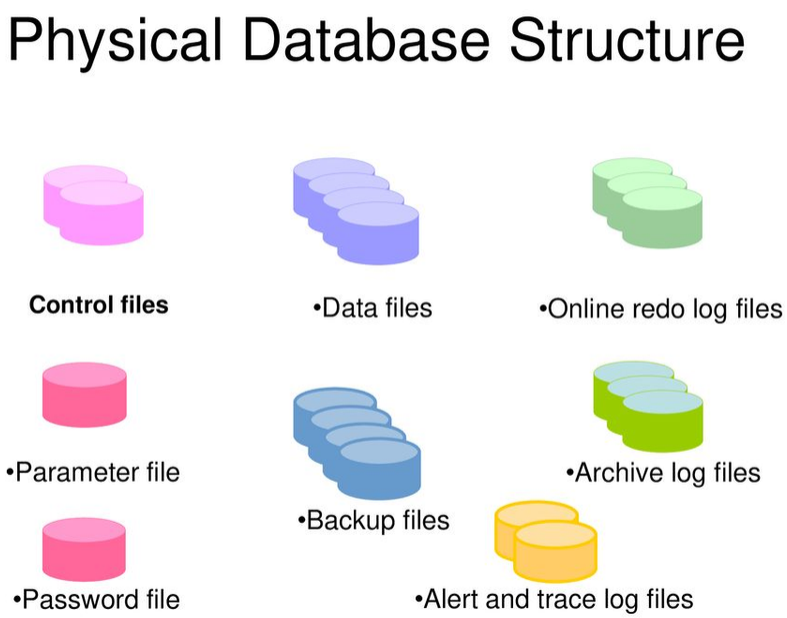
Нанася се новата информация директно в диска

Потребителят, който прави промяната вижда промените си веднага

Oracle Architectural components

Два основни компонента

- the **database** or the **physical structures**
- the **instance** or the **memory structures**



Всеки път, когато се стартира една БД, тя се нуждае от буфери, кешове, т.н.

Всяка БД идва със стандартна конфигурация, която може да се променя - нагласяне на буфери, големината им, всичките тези параметри са във Parameter file

1. **Control files**

   Съдържат информация за цялата структура на БД, къде какво се намира, разделени са на таблични пространства, сегменти, ...

   В контролните файлове винаги се съхраняват няколко екземпляра, за да се предотврати загубата на информация.

   Всеки път като се стартира БД се започва зареждането на данните, необходими за работа със съответните потребители и т.н.

2. **Redo log files**

   Record changes to

3. Segments, extents, and blocks
   a. segments exist within a tablespace
   b. segments consist of a collection of externs
   c. externs are a collection of data blocks
   d. data blocks are mapped to OS blocks

   Обикновено големината на тези блокове е число, кратно на блоковете в ОС

4. Tablespaces and data files

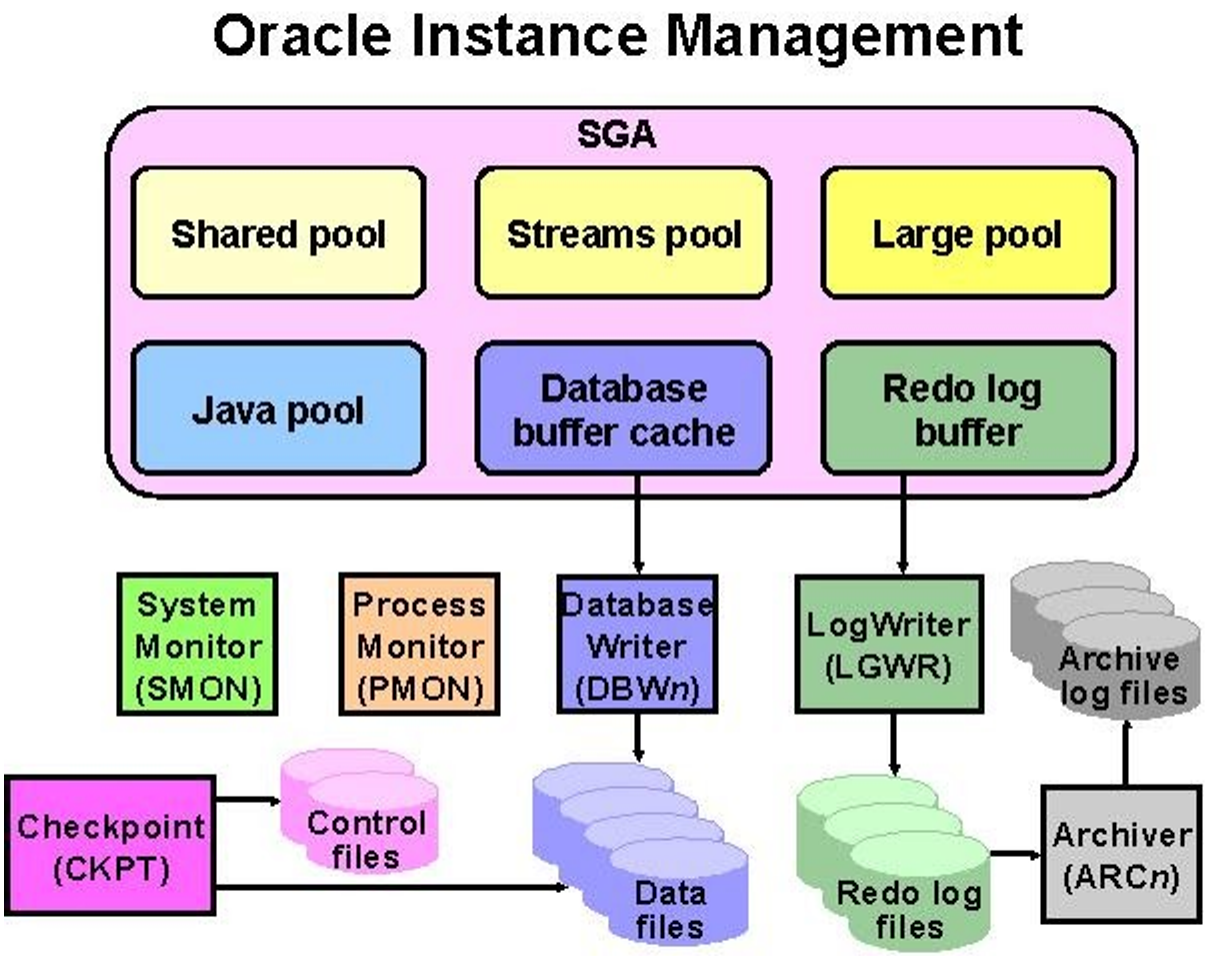
   В тези таблични пространства, oracle съхранява информацията

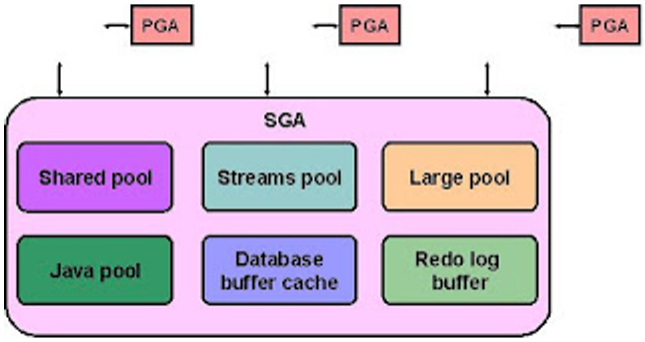
   Tablespaces consist of one or more data files

   data files belong to only one tablespace

   Всяка БД идва с едно системно таблично пространство, хубаво е да не ползваме него, а да създадем свое таблично пространство

   Като стартираме БД, имаме текущият екземпляр





Processing a SQL Statement

1. Connect to an instance using
    a. the user process
    b. the server process
2. The oracle server components that are used depend on the type of SQL statement
    a. queries return rows

        В резултат на заявка получаваме временна таблица - къде да бъде записан и колко време да се пази.

    b. DML statements log changes
    c. commit ensures transaction recovery
3. some oracle server components do not participate in SQL statement processing

Processing a Query

Да се провери дали съществува такава таблица, дали имаме право да достъпим тази информация

1. Parse
    a. search for identical statement