CS307

Database Principles

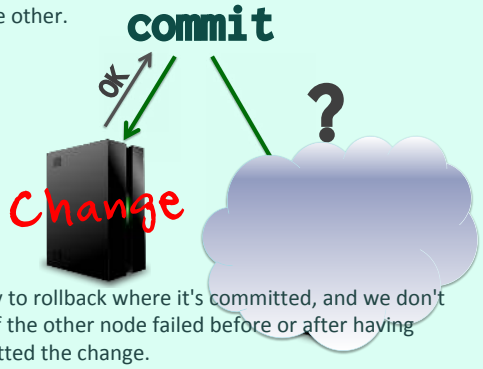
Stéphane Faroult
faroult@sustc.edu.cn

Liu Zijian liuzijian47@163.com

One big problem is with transactions that involve several servers. Remember that transactions are meant to be atomic operations (a principle challenged by NoSQL databases).

Distributed Transactions

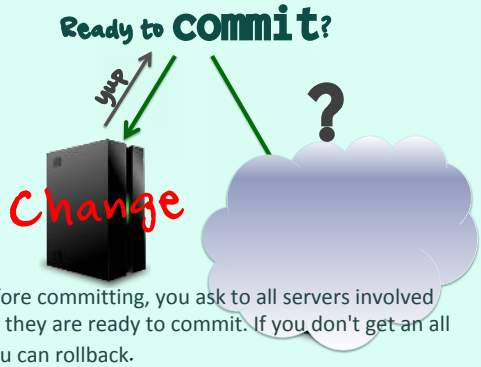
It may happen that when we commit we know for sure it worked on one server, and we get no acknowledgment from the other.



No way to rollback where it's committed, and we don't know if the other node failed before or after having committed the change.

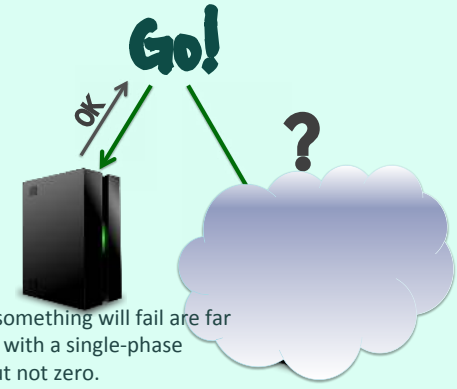
TWO-PHASE COMMIT

One algorithm was devised (a long time ago), called a "two-phase commit".



JUST before committing, you ask to all servers involved whether they are ready to commit. If you don't get an all clear, you can rollback.

Otherwise, you can then (2nd phase) send the official "commit" signal



Odds that something will fail are far lower than with a single-phase commit, but not zero.

There still may be IN-DOUBT transactions

You can still have cases when you don't know if one of the servers did or didn't commit in the end, and you end up with "in-doubt transactions" that you have to resolve (which means cancel) manually. It happens far more often than you may think, simply because the number of transactions in a big company is enormous, and a tiny percentage still means a few cases every week.

Latency

Additionally, you have latency issues. All machines in a cluster may not be sitting next to each other, they may be a few kilometers apart in different data centers for security reasons (fire, flood, typhoon, ...)

1 KM = 0.000005s

Even if information travels fast, multiplying exchanges (two-phase commit) may become a sensitive issue.

Synchronous? Asynchronous?

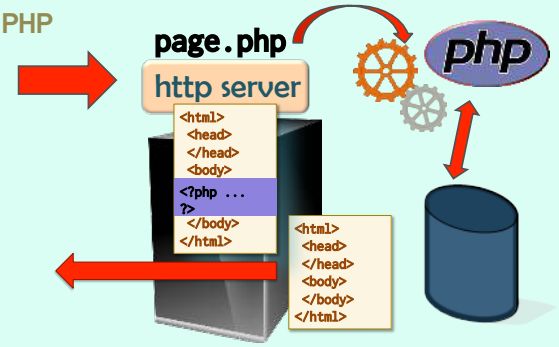
At which point you have to choose between a synchronous mode in which you always wait for an acknowledgment that everything went fine, or an asynchronous mode in which you cross fingers and switch to something else. The type of hardware you have may influence your decision: if all your disks have a big buffer and a reliable battery for instance, you may be more likely to trust them.

Programming with Databases

a PHP example

We have explored SQL inside out, we have seen how to code inside the database with functions, procedures and triggers, let's briefly (it's far easier) how to access a database from a program.

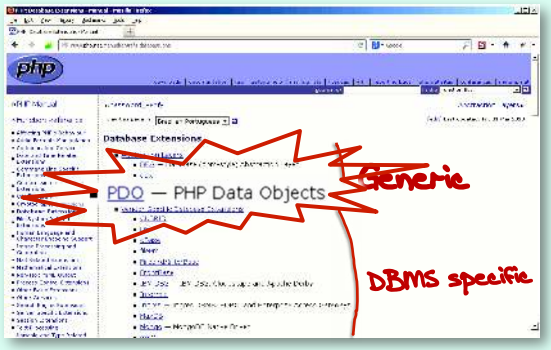
PHP



Let's take the example of PHP, a scripting language popular with websites.

PHP

PDO is a good choice.



PHP

```
<?php
   phpinfo();
?>
```

Tells you which DBMS products you can talk to. Try to find and install the driver if you don't have it.

PDO

PDO support	enabled
PDO drivers	dblib, mysql, odbc, pgsql, sqlite

pdo_dblib

PDO Driver for FreeTDS/Sybase DB-lib	enabled
Flavour	freetds

pdo_mysql

PDO Driver for MySQL	enabled
Client API version	5.5.30

Directive	Local Value	Master Value
pdo_mysql.default_socket	/var/lib/mysql/mysql.sock	/var/lib/mysql/mysql.sock

PHP

```
<?php
  if (isset($argv) && (count($argv) > 1)) {
     parse_str(implode('&', array_slice($argv, 1)), $_GET);
  }
```

```
bash$ php Example.php country=de
```

```
http://localhost/Example.php?country=de
```

A useful trick: the instructions above allow you to make command line parameters when PHP is run in a console look as if they were coming from a HTTP request. It makes debugging far easier because a wrong PHP script called through HTTP usually just displays a blank browser page.

PHP

```php
<?php
  if (isset($argv) && (count($argv) > 1)) {
     parse_str(implode('&', array_slice($argv, 1)), $_GET);
  }
  if (isset($_GET['country'])) {
     $country = $_GET['country'];
  } else {
     echo 'You must provide a country code' . PHP_EOL;
     exit;
  }
  $db = new PDO('mysql:host=localhost;dbname=moviedb',
                'moviedb',
                'SQLsuccess',
                array(PDO::MYSQL_ATTR_INIT_COMMAND=>"SET NAMES utf8"));
  if ($db === false) {
     echo 'Connection to database failed' . PHP_EOL;
     exit;
  }
```

DBMS specific

username

password

optional setup

PHP

```php
    $query = 'select title, year_released'
           . ' from movies'
           . ' where country = lower(:country)'
           . ' order by year_released';
    $stmt = $db->prepare($query);
    if (($stmt !== false)
        && $stmt->execute(array('country'=>$country))) {
        $film_list = $stmt->fetchAll(PDO::FETCH_ASSOC);
        $cnt = 0;
```

PHP supports several syntaxes for parameter placeholders (including a Java-like ?). Names are easier, and make the code more readable. I'm passing parameters directly to the `execute()` method, as an associative (= indexed by names) array, and asking for an associative array as result as well. `$film_list` is an array of rows (arrays themselves)

PHP

```php
    foreach ($film_list as $film) {
       if ($cnt == 0) {
          echo '<table>'
             . '<tr><th>Title</th><th>Year</th></tr>' . PHP_EOL;
       }
       $cnt++;
       echo '<tr class="' . ($cnt % 2 == 0 ? 'even' : 'odd') . '">'
          . '<td>' . $film['title'] . '</td>'
          . '<td>' . $film['year_released'] . '</td>'
          . '</tr>' . PHP_EOL;
    }
    if ($cnt > 0) {
       echo '</table>' . PHP_EOL;
    } else {
       echo '<span class="error">No movies found</span>';
    }
```

`$film` is one row

PHP

Error management, as unexciting as ever. Method `errorInfo()`, available both with database and statement objects, returns an array, the message is at index 2.

```php
  } else {
     if ($stmt === false) {
        $flop = $db->errorInfo();
     } else {
        $flop = $stmt->errorInfo();
     }
     echo '<span class="error">' . $flop[2] . '</span>';
  }
  unset($db);
?>
```

DATABASE INDEPENDENCE

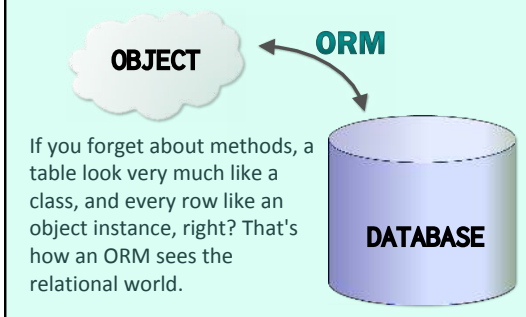
There are several dialects ...

If you want database independence, there is always the SQL dialect issue, functions that are all different, different NULL handling ... You must have something to reconstruct DBMS-specific SQL.

... let's add an abstraction layer !

Especially with Object Oriented programming, there is the idea of mapping objects to tables. Enters the ORM.

Object Relational Mapper



If you forget about methods, a table look very much like a class, and every row like an object instance, right? That's how an ORM sees the relational world.

django

Standard (more or less)

```
SELECT * from polls WHERE question LIKE 'Who%'
    AND (pub_date = '2005-05-02' OR pub_date = '2005-05-06')
```

Django specific

```
Poll.objects.get(
    Q(question__startswith='Who'),
    Q(pub_date=date(2005, 5, 2)) | Q(pub_date=date(2005, 5, 6))
)
```

As soon as things become just a little bit complicated, it all goes downhill. You end up with a complex SPECIFIC syntax even for simple queries (example from Django docs)

HIBERNATE

Escape route: HQL

Doesn't support WITH

Django is a Python framework (and a good one, for all I can say about the ORM). In the Java world, you have Hibernate. Where things become complicated, you can bypass ORM automation and use HQL - their watered-down, Greatest-Common-Divider SQL that doesn't support a lot of useful (and common) features.

In reality, among all the people who dream of database independence most of them won't migrate their applications to another DBMS. If they migrate, they'll also rewrite everything. If you want to minimise DBMS impact, it's far better to isolate YOUR SQL code than rely on something that is generic and often inefficient.

Isolate SQL code

Use DBMS features

Why pay for them otherwise?

Senior Database Administrator

.../...

What you will do: Recent job posting (US)

- Manage MySQL in production/QA/dev environments including installation, configuration, upgrades, schema changes, etc.
- Troubleshoot database issues, maintain database systems availability and scalability within production environments
- Perform capacity planning exercises to properly identify required hardware, software, database configuration/architecture necessary to support application needs
- Monitor database performance, identify performance problems and make adjustments to database parameters as needed.
- Monitor key performance indicators and make enhancements to improve/maintain performance/productivity at acceptable levels
- Enforce best practices for improving performance, scalability and operational manageability of production databases
- Part of on-call rotation to respond to and resolve application issues to ensure production applications are online

DBAs come in two flavors (in smaller companies, it's the same guy). They often have completely different backgrounds, and don't necessarily agree on everything.

PRODUCTION DBA

monitoring
shell scripting
security

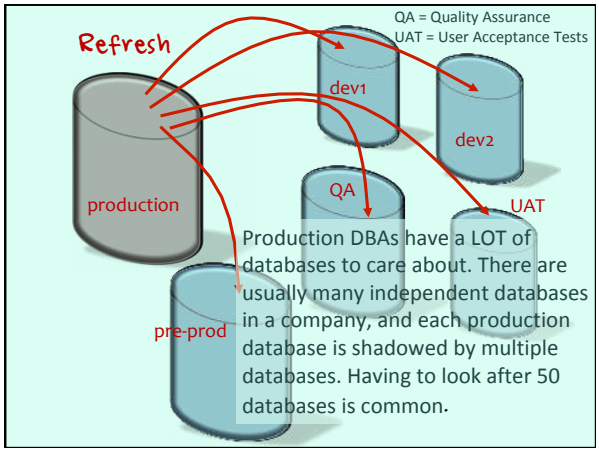
System Engineer

planning
architecture
SQL scripting

Developers

DEVELOPMENT DBA

In many ways, a development DBA is a developer who knows more than the average about databases. We'll focus here a little more on the job of a production DBA, because architects who design information system aren't always well aware of what their job is, and some "solutions" that look great on the paper are sometimes a hell to maintain in a daily production, with the constraints of keeping systems available and running as much as possible. Keeping everything simple and manageable should be the first concern of every architect.



Production DBAs have a LOT of databases to care about. There are usually many independent databases in a company, and each production database is shadowed by multiple databases. Having to look after 50 databases is common.

If you aren't square, you can't survive in this job. You must automate as many tasks as possible.

STANDARDS

ORGANIZATION

AUTOMATION

Other than performance issues (than we have partly seen) these are the big topics for a DBA. We have taked about privileges in labs, the other topics will be seen both in labs and lectures.

Physical Storage

User Management

Backup / Recovery