Выборка с агрегатами

Предложение GROUP BY используется для определения групп выходных строк, к которым могут применяться агрегатные функции (COUNT, MIN, MAX, AVG и SUM).

Группировка с условием

Если предложение WHERE определяет предикат для фильтрации строк, то предложение HAVING применяется после группировки для определения аналогичного предиката, фильтрующего группы по значениям агрегатных функций. Это предложение необходимо для проверки значений, которые получены с помощью агрегатной функции не из отдельных строк источника записей, определенного в предложении FROM, а из групп таких строк. Поэтому такая проверка не может содержаться в предложении WHERE.

Пример запроса GROUP BY

```
SELECT model, COUNT(model) as model_count, AVG(price) as
avg_price FROM auto
GROUP BY model
HAVING model_count > 1;
```

Взаимодействие PHP и mysql

MySQLi (MySQL Improved) — расширение драйвера реляционных СУБД, используемого в языке программирования PHP для предоставления доступа к базам данных MySQL.

Создание соединения

```
$mysqli = new mysqli('127.0.0.1', 'your_user', 'your_pass', '
sakila');
```

Выполнение запросов

```php
$selectQuery = 'SELECT field1, field2, field3 FROM table';

$resultQuery = $ourMysqli->query($selectQuery);
```

MEDIASOFT

Обработка результатов

```php
// получить данные одной строки в виде ассоциативного массива
$entry = $resultQuery->fetch_assoc();

// получить данные одной строки в виде объекта
$entry = $resultQuery->fetch_object();

// получить все строки, вариант № 1
$entries = array();
```

Подготовленные запросы

```php
if ($stmt = $mysqli->prepare("SELECT model FROM auto WHERE id=?")) {
    $stmt->bind_param("s", $id);
    $stmt->execute();
    $stmt->bind_result($model);
    $stmt->fetch();
    printf("%s в записи с id %s", $model, $id);
}
```

PDO

PDO - расширение для PHP, позволяющее работать с драйверами более 10 баз данных.

Создание подключения

```
$db = new PDO(‘mysql:host=localhost;dbname=testdb’,
$login, $passwd);
```

Выполнение запроса

```
$res = $db->query(“SELECT * FROM users”);
```

Подготовленные запросы

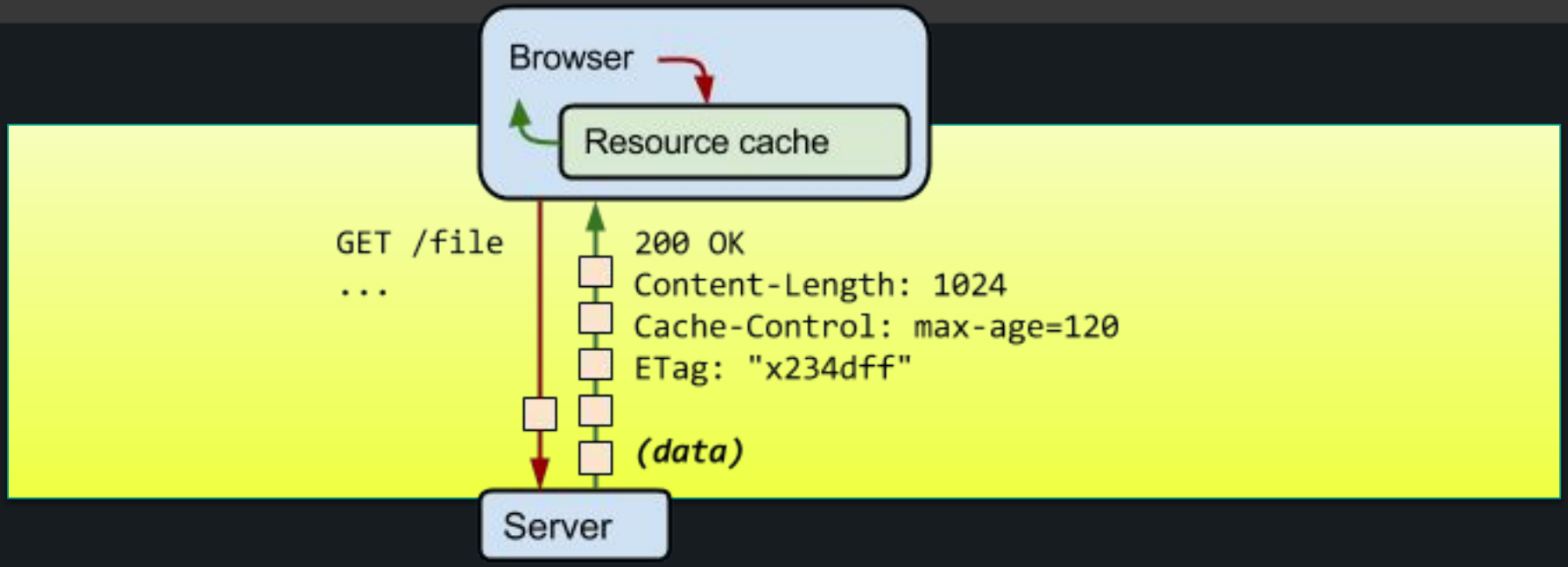
```
$db->prepare('SELECT * FROM users WHERE name=:name AND
email=:email');
$db->execute(array(':name'=>john,':email'=>'john@domain.com'));
$result = $db->fetchAll();
print_r($result);

$dbh->execute(array(':name'=>'alex',':email'=>'alex@domain.com'));
print_r($result);
```

Транзакции

Транзакция — это операция, состоящая из одного или нескольких запросов к базе данных. Суть транзакций — обеспечить корректное выполнение всех запросов в рамках одной транзакции, а так-же обеспечить механизм изоляции транзакций друг от друга для решения проблемы совместного доступа к данным.
Любая транзакция либо выполняется полностью, либо не выполняется вообще.

HTTP кэширование

Скачивать ресурсы страницы заново при каждом посещении - это очень неудобно. Из-за повторных отправок запроса сайт может работать медленно. Кроме того, пользователю придется зря тратить большое количество трафика. Именно поэтому кеширование данных имеет огромное значение при оптимизации сайта.


Browser
Resource cache
GET /file
...
200 OK
Content-Length: 1024
Cache-Control: max-age=120
ETag: "x234dff"
(data)
Server

Механизм работы кэша

Когда сервер возвращает запрос, он также отправляет набор HTTP-заголовков, описывающих тип контента, длину, команды для работы с кешем, маркер подтверждения и т. д. Например, в примере выше сервер возвращает запрос размером 1024 Б, отдает команду клиенту кешировать его на 120 секунд и отправляет маркер подтверждения (x234dff). Он используется, чтобы проверить, не изменился ли ресурс, после того как срок действия ответа истек.

Допустим, после нашего первого вызова прошло 120 секунд, и браузер начал новый запрос к тому же ресурсу. Сначала браузер проверяет локальный кеш и находит предыдущий ответ. Но его использовать нельзя, потому что срок его действия уже истек. Теперь браузер мог бы просто отправить новый запрос и получить ещё один полный ответ. Однако это неэффективно, потому что ресурс не изменился, и не имеет смысла снова скачивать байты, которые уже есть в кеше. Чтобы избежать этой проблемы, нужно использовать маркеры подтверждения, указанные в заголовках ETag. Сервер создает и возвращает произвольный маркер. Обычно это хеш или другая идентификационная метка файла. Клиент может не знать, как она создается, ему просто нужно отправить ее на сервер при следующем запросе. Если метка осталась прежней, то ресурс не изменился и скачивать его не надо.