3.ワークショップ
Geniee Speedup Challenge

タイムテーブル

休憩を含め5時間の長丁場です。

(といっても時間は思ったより少ないので

得意領域など**できるところから** 取り組みましょう！ ！ ！)

00:00:00 ~ 00:30:00	問題説明 + 環境確認
00:30:00 ~ 02:30:00	実装パート1
02:30:00 ~ 02:40:00	休憩 + 一次評価 + 参加者同士で相談可
02:40:00 ~ 05:10:00	実装パート2 (+ ヒント公開)
05:10:00 ~ 05:25:00	コード提出
05:25:00 ~ 05:40:00	解説 + 簡単な作業

広告配信の概要

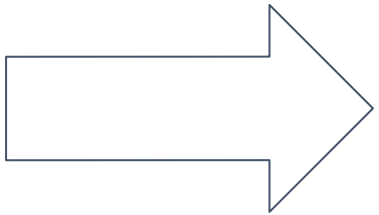
広告配信の概要

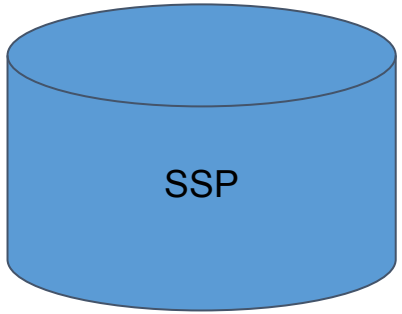
1. ユーザがWebページにアクセス

広告配信の概要

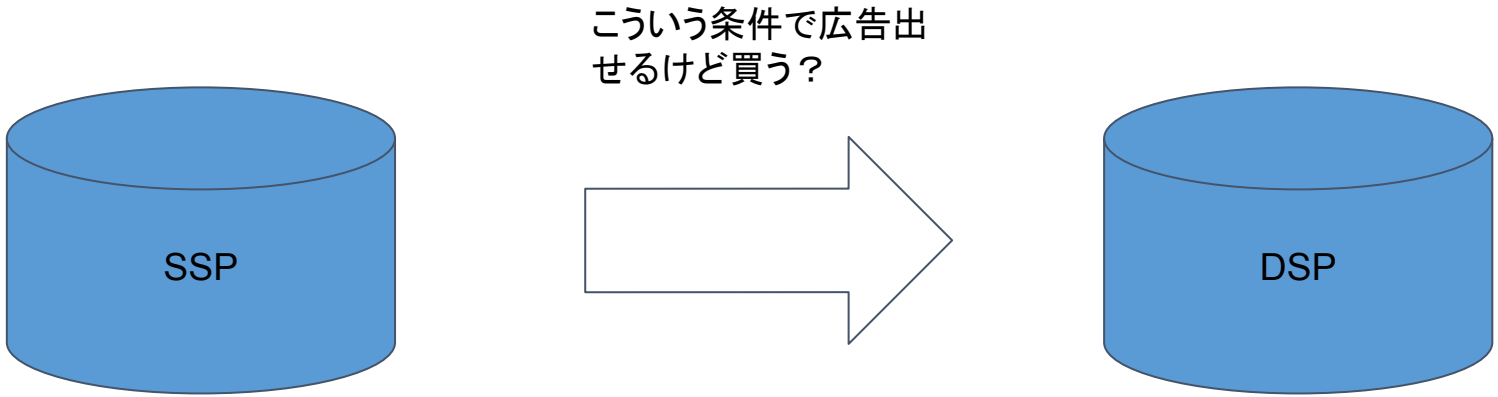
2. サイトに貼られている広告タグがSSPにリクエストを送信

サイト見られたよ



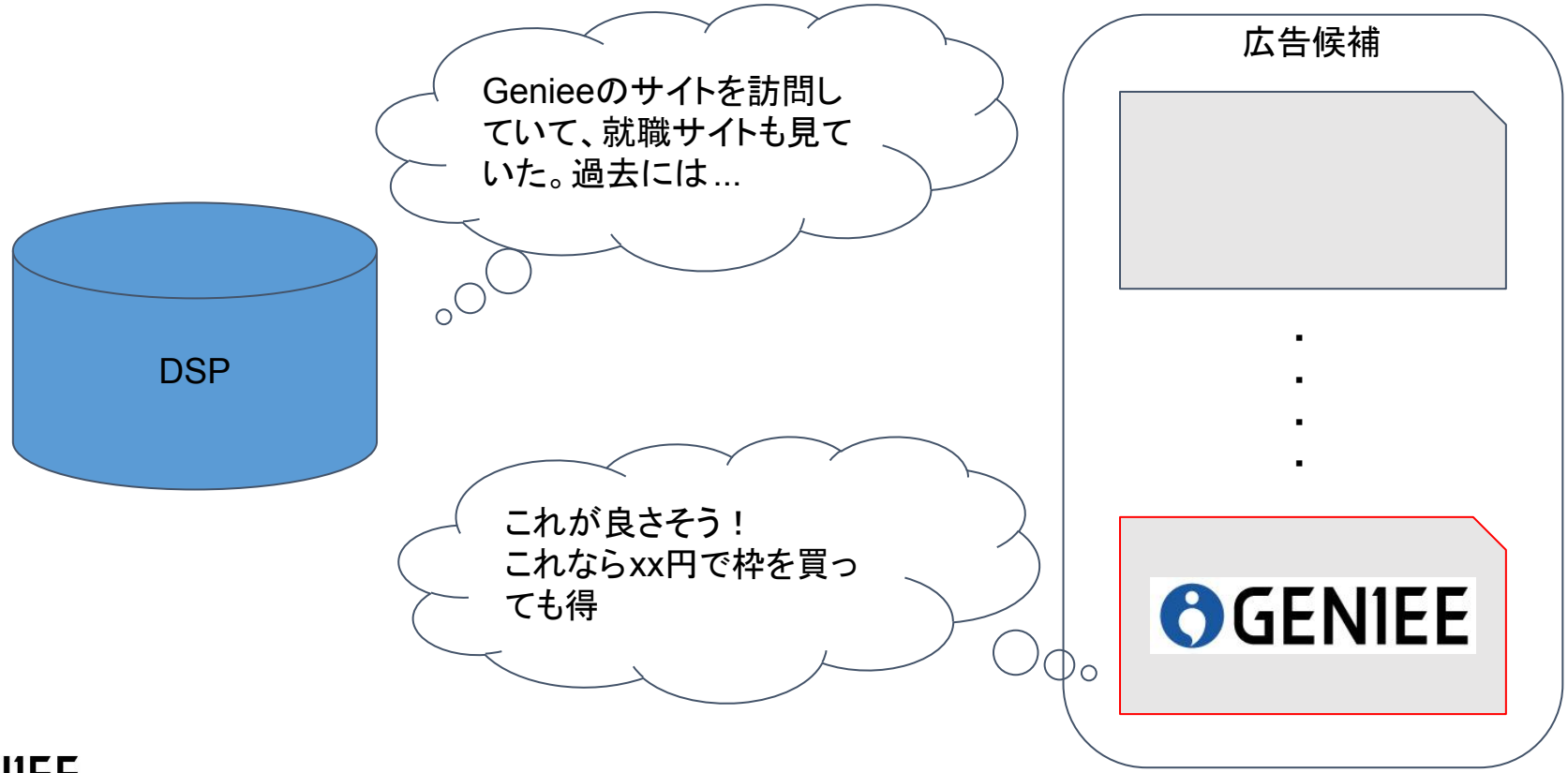


3. SSPがDSPに広告枠を売りに出す



広告配信の概要

4. DSPが数ある広告候補から適切なものと価格を決定



広告配信の概要

5. DSPはSSPに広告と入札価格を返却しサイトに広告が表示される

これを100ms ~ 300ms以内で行う

SSP

DSP

買い手がいたので
この広告を表示します

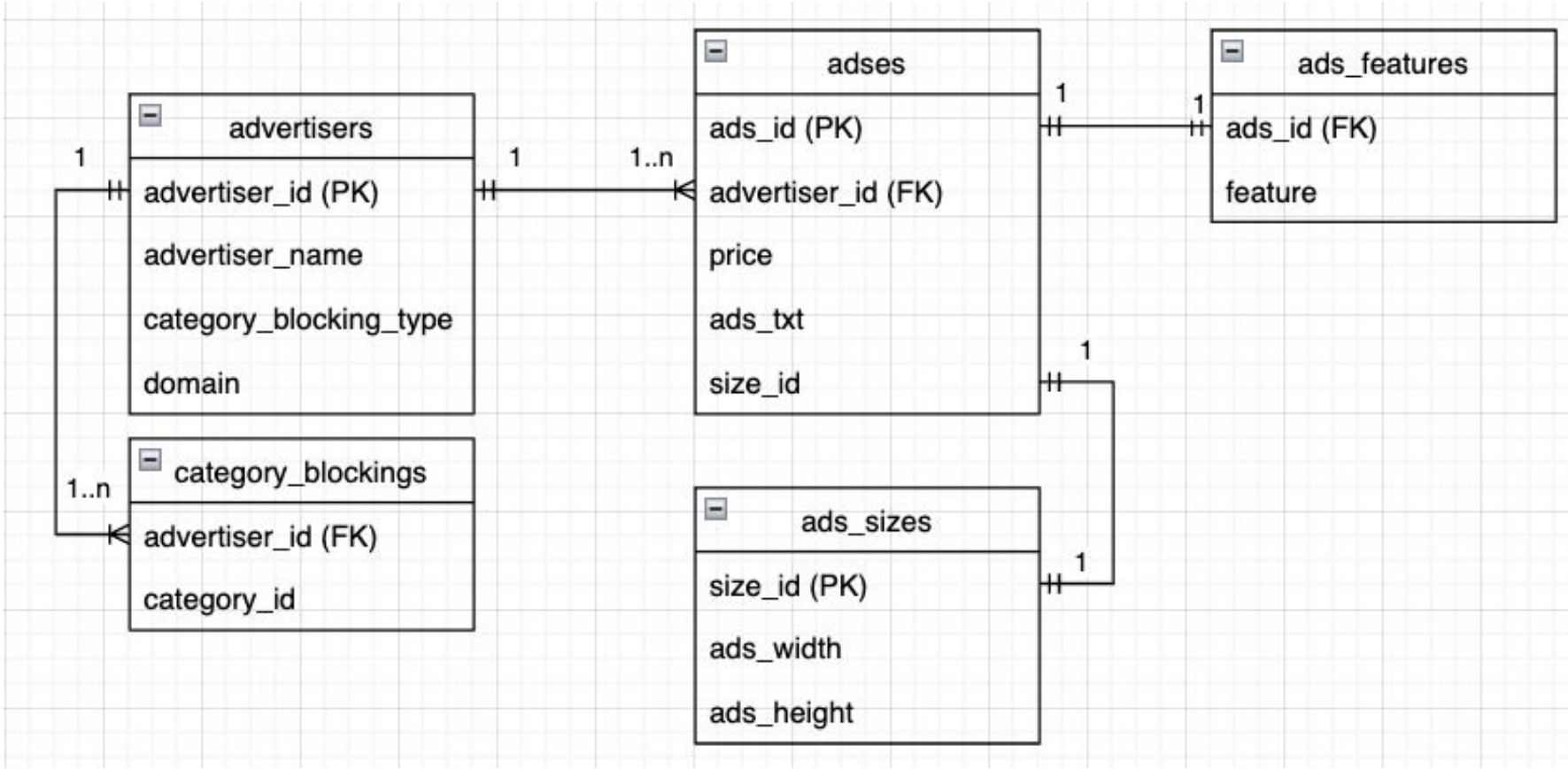
広告枠をxx円で買うので
この広告を出して！

題材発表

みなさんには...

簡易版DSPを高速化してもらいます

データの関連性（READMEにも画像があります）



簡易版DSPの要件

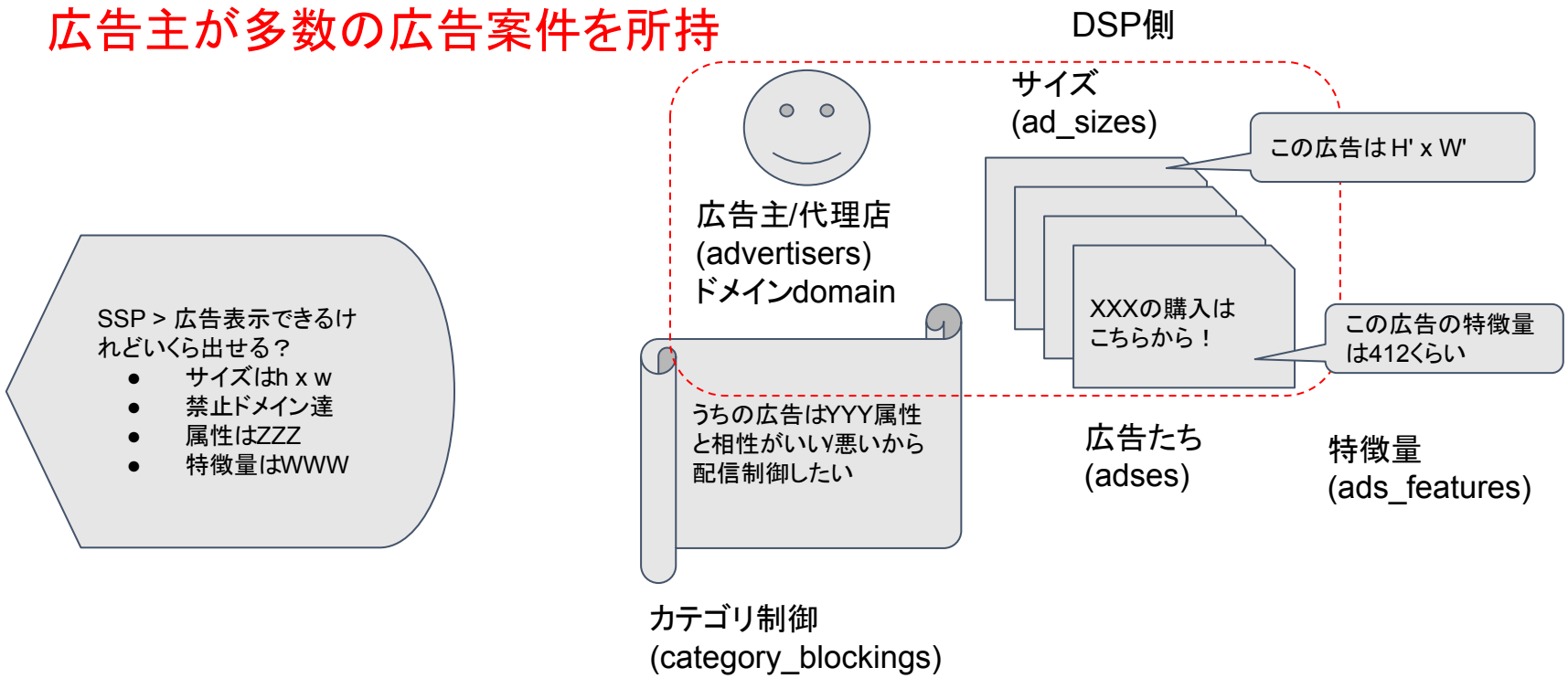
- 広告主が多数の広告案件を所持
- 要求広告サイズに合致するサイズの広告のみ抽出
- 広告ごとに広告主の拒否ドメインが存在
- 広告主ごとに配信許可/拒否カテゴリが存在
  - 教育・就職・書籍・スポーツなどなど
  - 枠にもカテゴリが存在し、枠と広告案件とで配信制御が必要
- 広告案件ごとに特徴量が存在
  - 枠にも特徴量が存在
  - 広告主ごとに最も特徴量にマッチしている1つのみが候補
- 候補となる広告の中で価格が最も高いもののみを返す

詳しくは既存実装およびREADMEを参照のこと

GENIEE

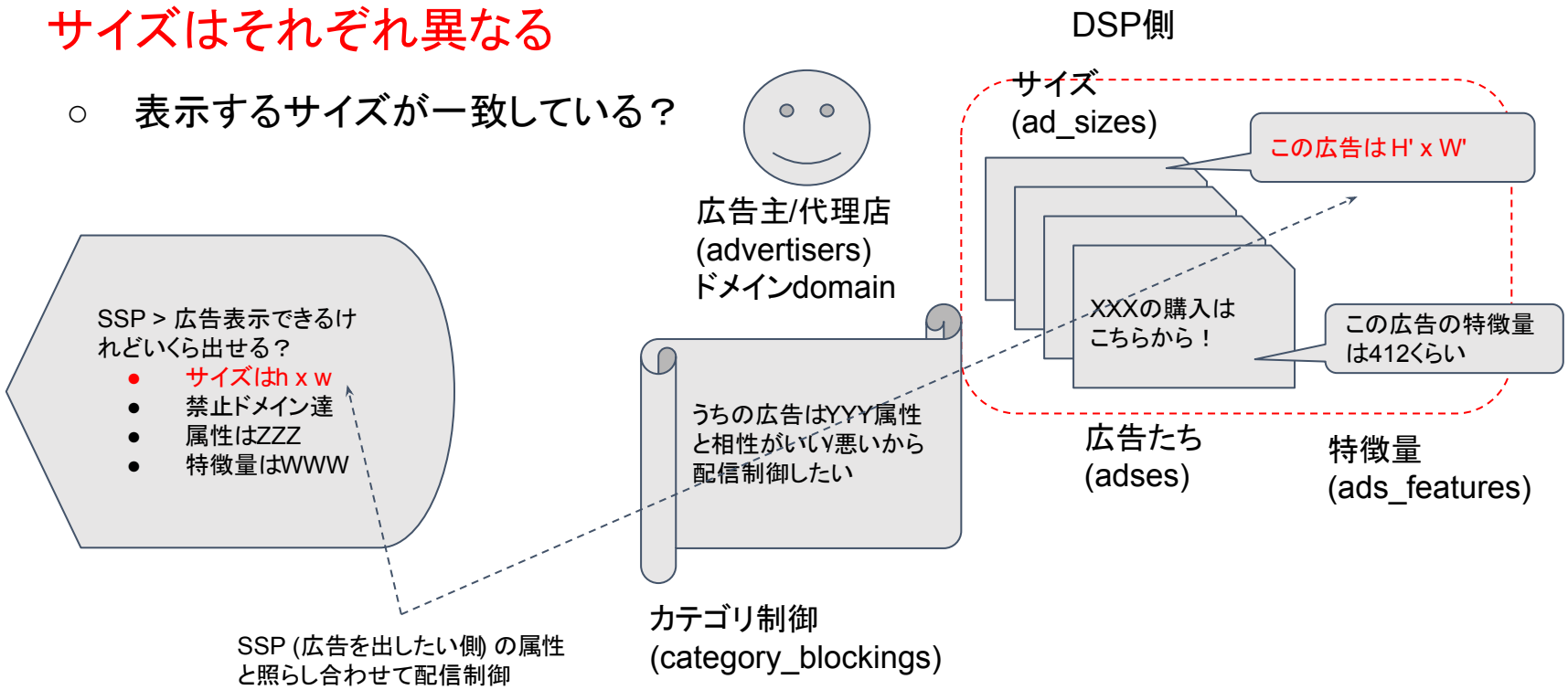
簡易版DSPのデータ関係性

- 広告主が多数の広告案件を所持



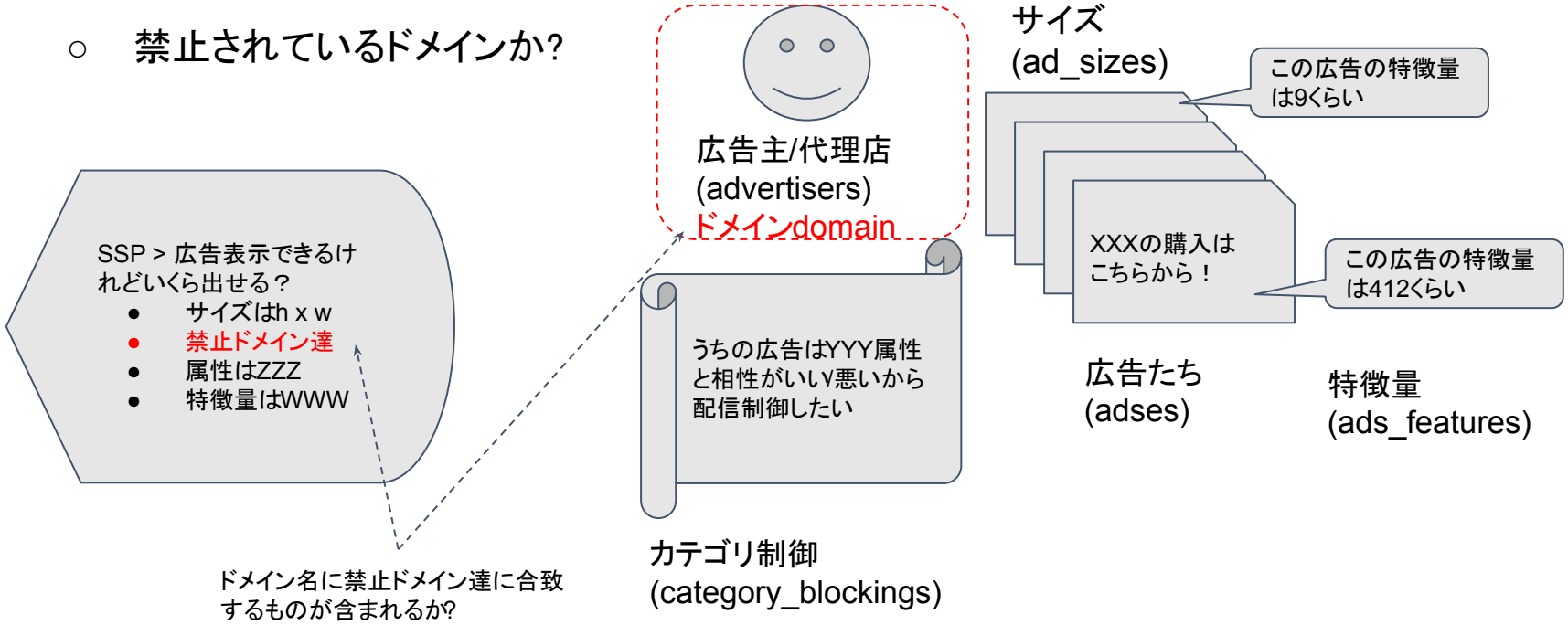
簡易版DSPのデータ関係性

- サイズはそれぞれ異なる
  - 表示するサイズが一致している？



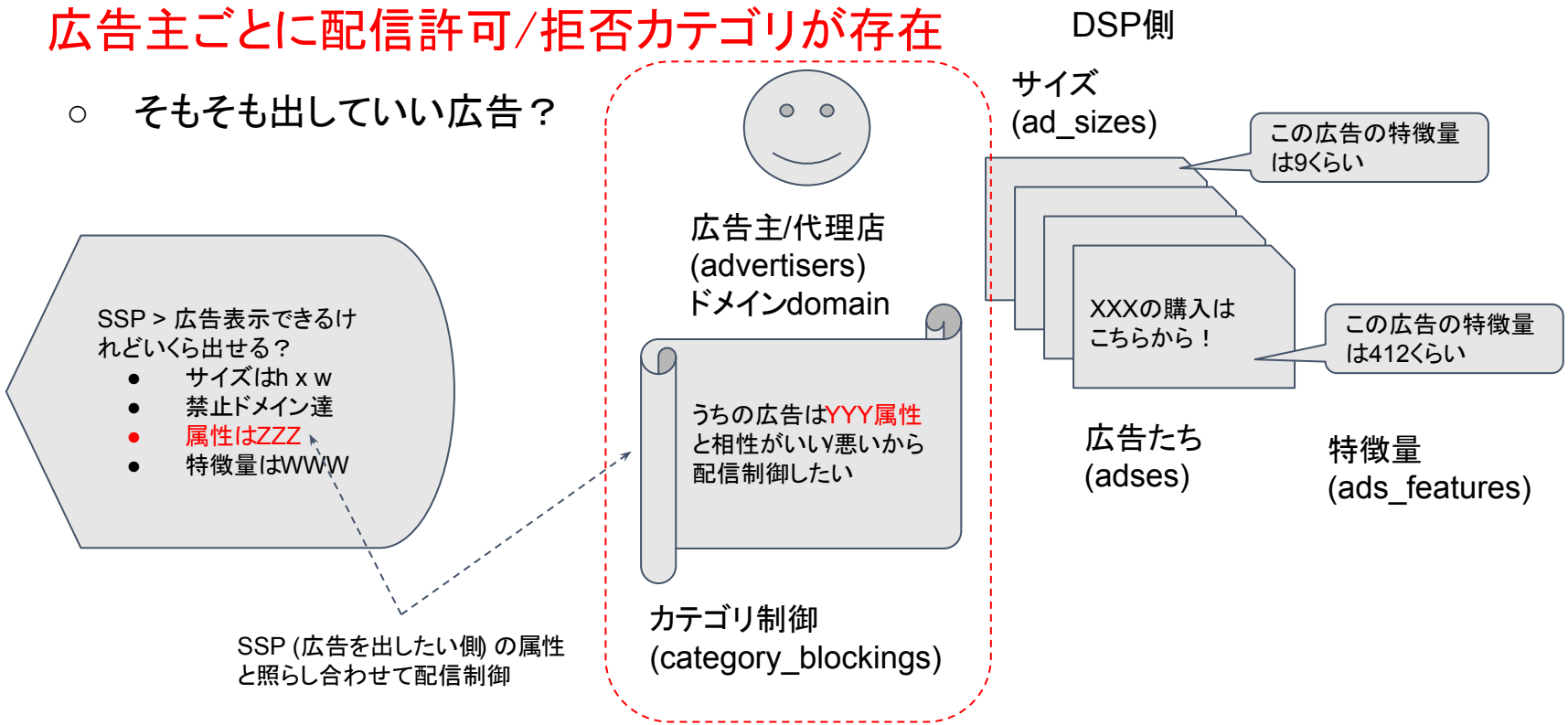
簡易版DSPのデータ関係性

- 広告主ごとにドメインが存在
  - 禁止されているドメインか?



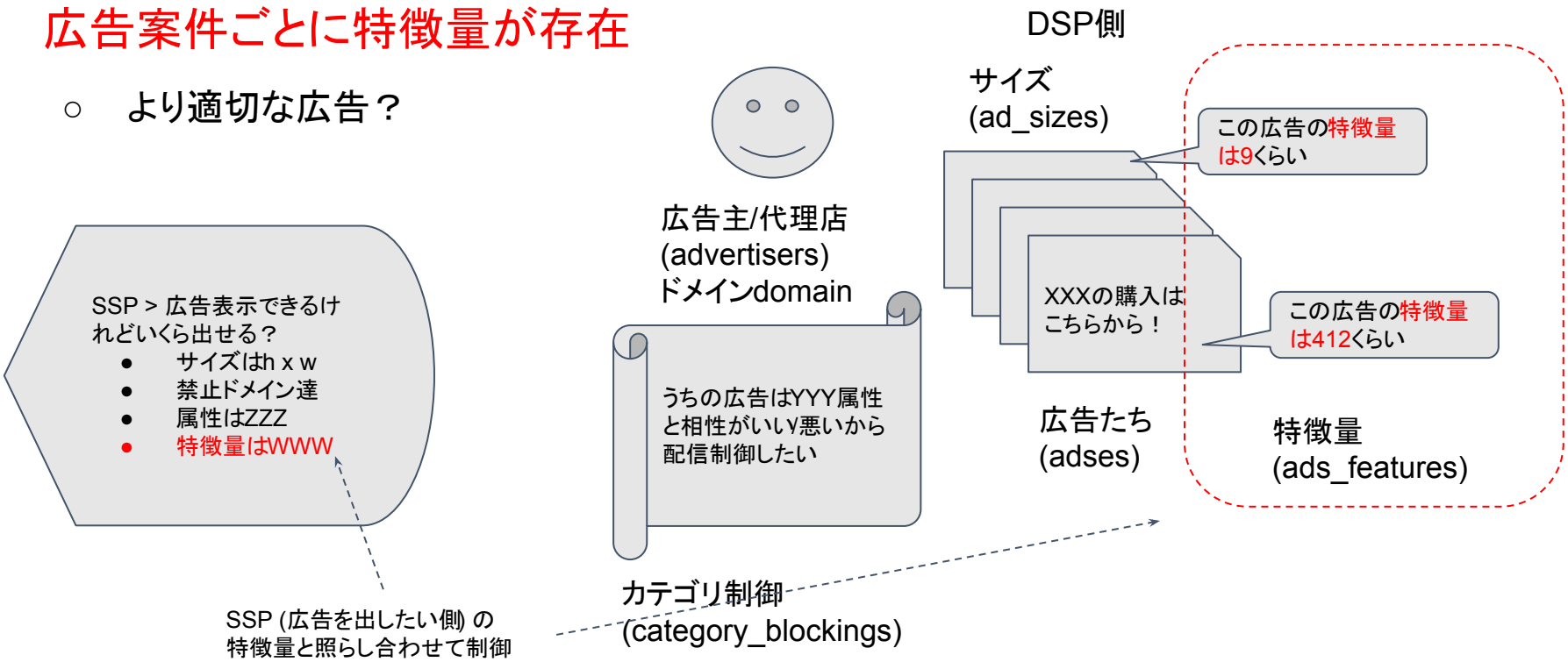
簡易版DSPのデータ関係性

- 広告主ごとに配信許可/拒否カテゴリが存在
  - そもそも出していい広告？



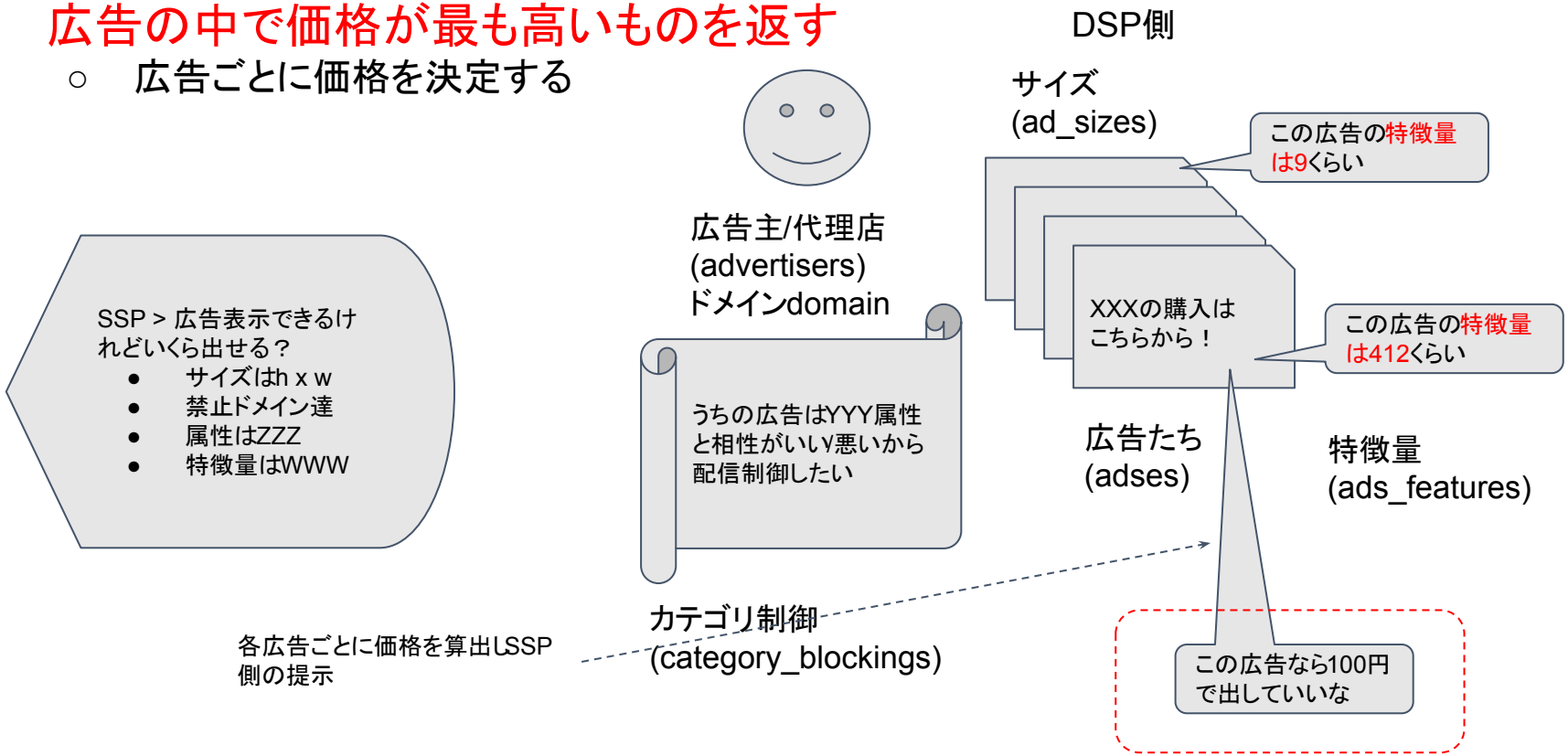
簡易版DSPのデータ関係性

- 広告案件ごとに特徴量が存在
  - より適切な広告？



簡易版DSPのデータ関係性

- 広告の中で価格が最も高いものを返す
  - 広告ごとに価格を決定する



具体例その1

`advertisers` は、広告主ID / 広告主名 / カテゴリ制御タイプ/ドメイン名を管理

```
mysql> select * from advertisers where advertiser_id = 1;
+---------------+-----------------+--------------------------+-----------------------------------+
| advertiser_id | advertiser_name | category_blocking_type | domain                            |
+---------------+-----------------+--------------------------+-----------------------------------+
|             1 | tdRTFlB5MRBV8SkZ |                      0 | K789YZWFinalCodeXYZ000CXYZ456YZ0.com |
+---------------+-----------------+--------------------------+-----------------------------------+
1 row in set (0.01 sec)
```

`adses` は、広告ID / 広告所有者の広告主ID / 価格 / 広告文字列/サイズIDを管理

```
mysql> select * from adses where size_id = 1 limit 1;
+--------+---------------+---------+-----------------------------------------------------------------------------------------+---------+
| ads_id | advertiser_id | price   | ads_txt                                                                                 | size_id |
+--------+---------------+---------+-----------------------------------------------------------------------------------------+---------+
|      2 |             1 | 38.4662 | 6d2ELfjwwRNMNTmzlXuj9BQvyk3U32BmYa2wUDFb9E89IFEcsVyOYqkFxDvnXnAT |       1 |
+--------+---------------+---------+-----------------------------------------------------------------------------------------+---------+
```

具体例その2

`ads_features` は広告ID / 広告特徴量を管理

```
mysql> select * from ads_features where ads_id = 1;
+--------+---------+
| ads_id | feature |
+--------+---------+
|      1 |  -94368 |
+--------+---------+
1 row in set (0.00 sec)
```

`category_blockings` は広告主ID / カテゴリIDを複数レコードで管理

```
mysql> select * from category_blockings where advertiser_id = 1;
+---------------+-------------+
| advertiser_id | category_id |
+---------------+-------------+
|             1 |           3 |
|             1 |          19 |
+---------------+-------------+
2 rows in set (0.00 sec)
```

GENIEE

具体例その3

`ads_sizes` は、サイズID / 広告の横幅 / 広告の縦幅を管理

```
mysql> select * from ads_sizes limit 2;
+---------+-----------+------------+
| size_id | ads_width | ads_height |
+---------+-----------+------------+
|       1 |       200 |        300 |
|       2 |       250 |        350 |
+---------+-----------+------------+
2 rows in set (0.00 sec)
```

`adses`（再掲）

```
mysql> select * from adses where size_id = 1 limit 1;
+--------+---------------+---------+------------------------------------------------------------------------------------------+---------+
| ads_id | advertiser_id | price   | ads_txt                                                                                  | size_id |
+--------+---------------+---------+------------------------------------------------------------------------------------------+---------+
|      2 |             1 | 38.4662 | 6d2ELfjwwRNMNTmzlXuj9BQvyk3U32BmYa2wUDFb9E89IFEcsVyOYqkFxDvnXnAT |       1 |
+--------+---------------+---------+------------------------------------------------------------------------------------------+---------+
```

具体例その3

リクエストには

- オークション識別子 (`auction_id`)
- 広告枠のカテゴリ情報 (`category_ids`)
- 枠の特徴量 (`zone_feature`)
- 広告の横幅 (`zone_width`)
- 広告の縦幅 (`zone_height`)
- 禁止ドメインリスト(`block_domains`)

の情報が送られてくる

これらの情報で広告を選択していく

具体例その4

広告サイズの制御（サイズの範囲: 横幅、縦幅ともに1 ~ 999）

例えば要求広告サイズが(横幅, 縦幅) = `(100, 200)` の場合、

広告データのうち、

(横幅, 縦幅) = `(100, 200)`の広告のみがオークション対象候補に選ばれる

横幅が異なる、もしくは縦幅が異なる広告は選ばれない

具体例その5

禁止ドメインによるブロック

- 広告主ドメインは32文字の英大文字小文字数字+.comからなる文字列
- 禁止ドメインは3~24文字長で英大文字小文字数字から構成される

例えば禁止ドメインが (prohibit, test,hoge) の場合、

広告主ドメインで少なくとも一つが含まれていたら選ばれない。

広告主ドメイン例)

❌ hogefuga.com , googletest.com , dprohibitptestg.com

⭕ googletes.com, fhoguga.com, geniee.com

GENIEE

具体例その6

カテゴリ配信制御 (IDの範囲: 1 ~ 20)

例えば広告主がカテゴリID = `{3, 19}` をブロックしたいとする

その時リクエストのカテゴリID = `{1, 3, 5, 8, 14}` だとすると、

`3` がブロックしたいIDに含まれている

→ この広告主の広告は配信されない

具体例その7

特徴量ターゲティング（特徴量の範囲: -100000 ~ 100000）

特定の広告主が所有している広告の特徴量が

`(-94368, 12949, 4043, 20375, 42454)` であるとする

リクエストの特徴量が `31415` だとすると、差分が

`(125783, 18466, 27372, 11040, 11039)` となるので

最右の広告が選択される

環境整備

0.まずPCにログイン

ID : geniee

PASS : qwer2025

1. GitHubを確認

- デスクトップに以下のgitをクローンしたものが置いてあります
  - web上で見たい方は中のgit-web.weblocを実行すると良いです

URL: https://github.com/kiyoshi0205/geniee-backend-intern.git

以降の手順はスライドにもGitHub上のREADMEにも記載されています

2. 環境構築

- ターミナルを開く or vscodeを開いて以下を実行

```
~/geniee-backend-intern/setup/setup-server.sh
```

  - 適度にtabキーで補完しながら実行すると良いです

- エディタを開いて開発環境に移動

```
code ~/geniee-backend-intern
```

コンパイル && 検証コマンド実行

コンパイル

プログラムを編集したあと、(コンパイル言語を選択した方は)実行形式にするためにコンパイルする必要があります。コマンドは以下です。

```
user@mac$ make build
```

実行ファイルとして `auction-worker` が生成されます。

実行

必要に応じてコンパイルをしたのち、以下のコマンドを叩くと各言語に応じたサーバが起動します。デバッグや動作確認にお使いください。(`make run` 時に現在起動しているプロセスを停止するため、複数起動できないことに注意してください)

```
user@mac$ make run
```

初期テスト結果

みなさんのPCで試してみた結果

C++だと19.5sec, pythonだと10sec、golangだと3secくらい

```
----------------------small----------------------
init DB data...   done.
run auction-worker...
wait 5 second for setup auction worker...
[####################] 10/10 (0.51 req/sec)
response check PASSED
elapsed time: 19.4622 sec. performance check Time Limit Exceeded...
FAILED...
make: *** [Makefile:60: inner-check] Error 1
make: Leaving directory '/tmp/intern'
make: *** [check] Error 2
```

スピードテストで行っている内容

1. DBの初期化
2. すでにコンパイル/実行準備されている `auction-worker` の実行
3. 5秒間待つ
    - この間にauction-workerの下準備ができます
4. パフォーマンス測定
    - 10並列でリクエストを投げます
5. auction-workerの終了

毎回 `auction-worker` を起動しなおしていることに注意

評価方法

規模の異なるDB/リクエストを飛ばし速度を競います

規模	広告主数	広告案件数 per 広告主	サイズ数	カテゴリID数	リクエスト数	検証閾値
small	20~30	20~30	4	1~10	10	2sec
medium	300~450	300~450	4	1~10	100	5sec
large	600~900	600~900	4	1~10	500	5sec
challenge	600~900	600~900	4	1~10	5000	10sec

リクエスト数分送られるデータを検証閾値秒以内で処理してください

GENIEE

ちなみに...

GenieeのDSPでは一日20億くらいリクエストが飛んできています

PCのスペックや並列数にもよるが 0.50req/sec では捌ききれない

Challengeより規模の大きいデータセットを最低でも 20req/sec くらいで捌きたい

（本番ではもっと複雑なことをしているので数値は下がりますが...）

Gitの設定をしよう（任意）

開発過程でGitブランチを切りたいこともあると思います。仮想環境内で

```
git config user.name '名前をアルファベットで'
git config user.email '名前の後に@example.com'
```

の設定をしましょう(pushしないので適当でもOK)。下は例

```
git config user.name  'ryota-komiyama'
git config user.email  'ryota-komiyama@example.com'
```

サンプル実装

初期実装では3種類の言語で実装されています

- C++ (C++20; GCC 12.3.0)
- Python (3.12.5)
- Golang (1.22.2)

全ての言語においてChallengeをクリアできることは確認済みです

ただし、難易度が同じとは限りません。

(易) Golang < C++ < Python (難)

の順で難しくなります。ぜひ得意な言語で挑戦してください！

まずは言語選択をしよう

お好きなエディタでMakefileを編集し、6 ~ 8行目を使用したい言語に
変更してください。

例えばpythonを使用したい場合は、6行目をコメントアウトして7行目の `#` を削除してください。

```
3    SEED = 123
4
5
6    LANGUAGE = cpp
7    # LANGUAGE = python
8    # LANGUAGE = golang
9
```

参考までに…

SQLはどのくらいわかりますか？

1. 存在を知らない
2. 存在は知っていて、SELECTなど簡単なクエリなら読めるが書けない
3. SELECTなど簡単なクエリの読み書きはできる
4. DML全般把握しており、複雑なクエリも書ける
5. DML/DDL把握しており、正規化を意識したテーブル設計もできる

ちなみにみなさんに提供しているのはMySQL8.0です

実装スタート

皆さんと勝負しましょう！

- `make submit` を実行すると別seedによる結果を順位表に反映
  - 氏名はPC名なので安心してください
    - test/submit_user.txtを弄ると名前は変更可能です(変えても良いけど常識的な範囲で…)

順位表

#	氏名	Small	Medium	Large	Challenge
1	ryota-komiyama	116ms	159ms	334ms	2334ms
2	J0050	112ms	177ms	374ms	2680ms

提出履歴

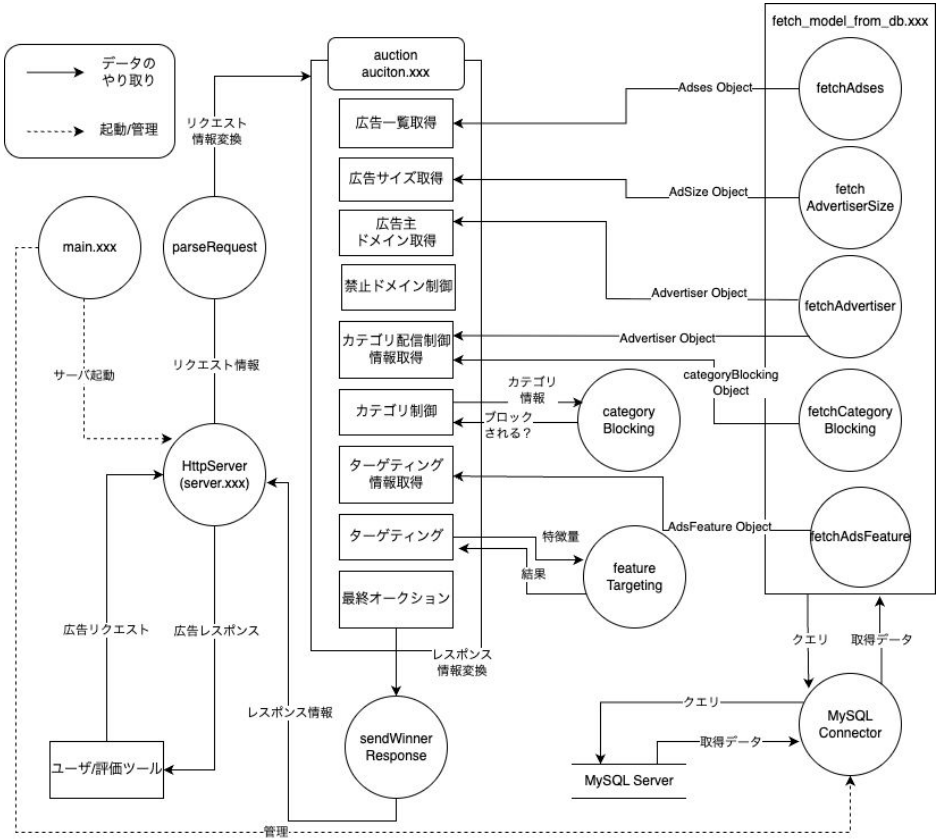
Name	Dataset	Language	Status	Time
J0050	Challenge	C++	AC	2680ms
J0050	Large	C++	AC	374ms
J0050	Medium	C++	AC	177ms
J0050	Small	C++	AC	112ms

実装

- まずはコードを理解しよう
  - auctionやfetch_model_from_dbあたりから理解しましょう
  - READMEに仕様詳細とソースコードの説明がありますが
    仕様理解はmedium突破には必須情報ではありません！
- `src/` 配下のコードをいじってみよう
  - 高速化を試し `make build` → `make check` の繰り返し
  - ある程度完成したら `make submit` で提出
- 以下の場所は編集してはいけません
  - `test/` 、`data/` 配下

サンプルコードの呼び出し階層（READMEにも画像があります）

1. main関数がserverを起動
2. リクエストが来たらserverがauctionを呼び出す
3. MySQLConnectorを通じて広告一覧を取得
4. データを取得しサイズ制御
5. データを取得しドメイン制御
6. データを取得しカテゴリ制御
7. データを取得しターゲティング
8. 最終オークションで広告選択
9. 結果を呼び出し側に返却



FAQ

FAQ

検証テストのレスポンスチェックが失敗した

検証バッチはレスポンスを `./test/result.json` に吐き出したのち、`auction_id` でソートします。
そのため例えばsmallデータセットを用いた場合、

```
user@mac$ diff ~/geniee-backend-intern/data/response_123_small.json ~/geniee-backend-intern/test/result.json
```

のように実行することで、どのデータがずれていたか確認することができます。

コンテナが固まった

コンテナをリスタートします。

`./setup/restart-server.sh` を実行してください。

```
user@mac$ ~/geniee-backend-intern/setup/restart-server.sh
```

各種インストールしたライブラリや設定は基本的には保存されたままです。

FAQ

検証テストのレスポンスチェックが失敗した

検証バッチはレスポンスを `./test/result.json` に吐き出したのち、`auction_id` でソートします。
そのため例えばsmallデータセットを用いた場合、

```
user@mac$ diff ~/geniee-speedup-2024/data/response_123_small.json ~/geniee-speedup-2024/test/result.json
```

のように実行することで、どのデータがずれていたか確認することができます。

仮想環境が固まった

仮想環境をリスタートします。

`./setup/restart-server.sh` を実行してください。

```
user@mac$ ~/geniee-speedup-2024/setup/restart-server.sh
```

上記コマンドを実行すると再度仮想環境に入った状態でvscodeの画面が起動します。

各種インストールしたライブラリや設定は基本的には保存されたままです。

早引きFAQ（READMEの掲載箇所早見表）

- 単体実行したい
  - サーバの立ち上げ: `doc/command.md#実行`
  - リクエストを投げる: `doc/command.md#単体の検証`

- プログラムFailedしたので調査したい
  - 正解との比較方法: `doc/faq.md#検証テストのレスポンスチェックが失敗した`
  - `make check` 中の出力が見たい: `doc/command.md#検証`

- 詳細な仕様が知りたい
  - featureTargeting: `doc/feature-targeting.md`
  - categoryBlocking: `doc/category-blocking.md`

生成AIの使用は許可されています

ChatGPT / Copilot / etc… お好きなものをお使いください

（Writerも使って確認しております）

コードを生成してもらってもよし、アドバイスをもらってもよし

業務でもうまく活用して効率的に働くことが求められるので是非

（実は使われること前提でちょっと難しくしています）