

# Движение ML алгоритмов в ПРОМ

Дмитрий Бугайченко



**СБЕРБАНК**

*Всегда рядом*

# Интеграция с «продом»

---

# Уровни интеграции

- Экспорт результатов
  - Используется тот же код
  - Не подходит для быстрого вычисления
  - Лишняя работа для тех, кто не пришел
- Экспорт моделей
  - Быстрая реакция
  - Жесткие требования по производительности
  - Расхождения между тренировкой и эксплуатацией
  - Модель тоже устаревает
- Асинхронное дообучение

# Экспорт моделей: варианты



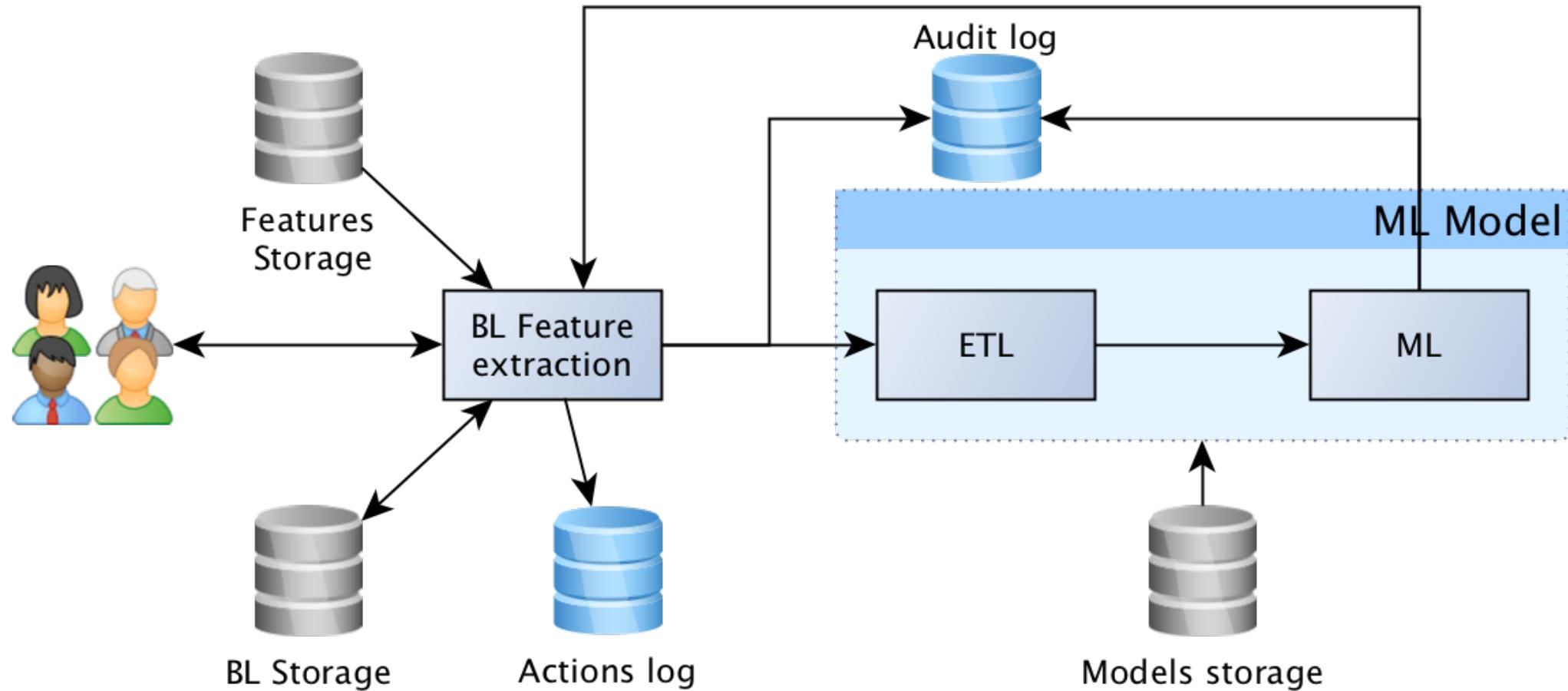
- «Стандартный формат»
- Есть интерпретаторы:  
JPML
- Спарк сохраняется в  
PMML

- Поддерживает  
Spark/Sklearn/TF
- Переваривает в свой  
«бандл»

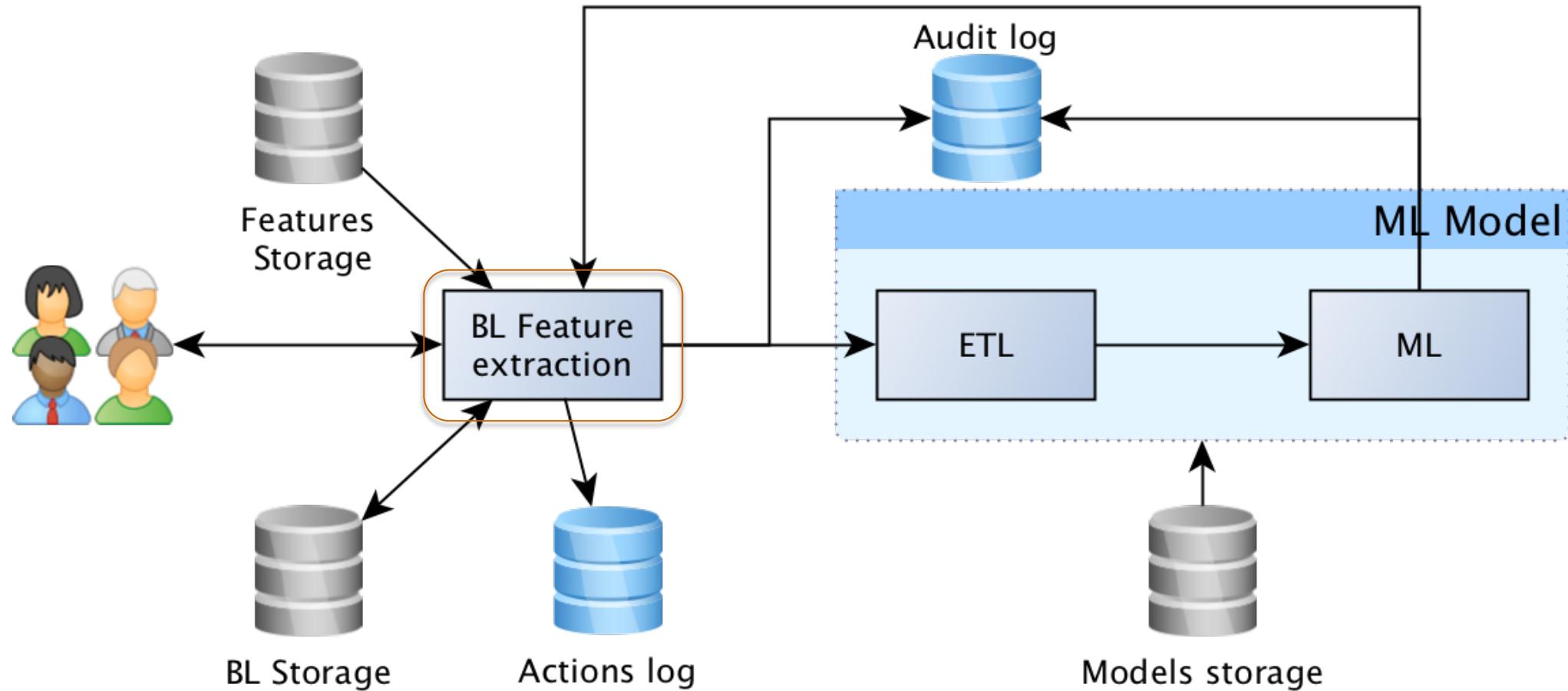
# Экспорт моделей: варианты

- Spark Local
- Spark Streaming Local
- PFA
- ...

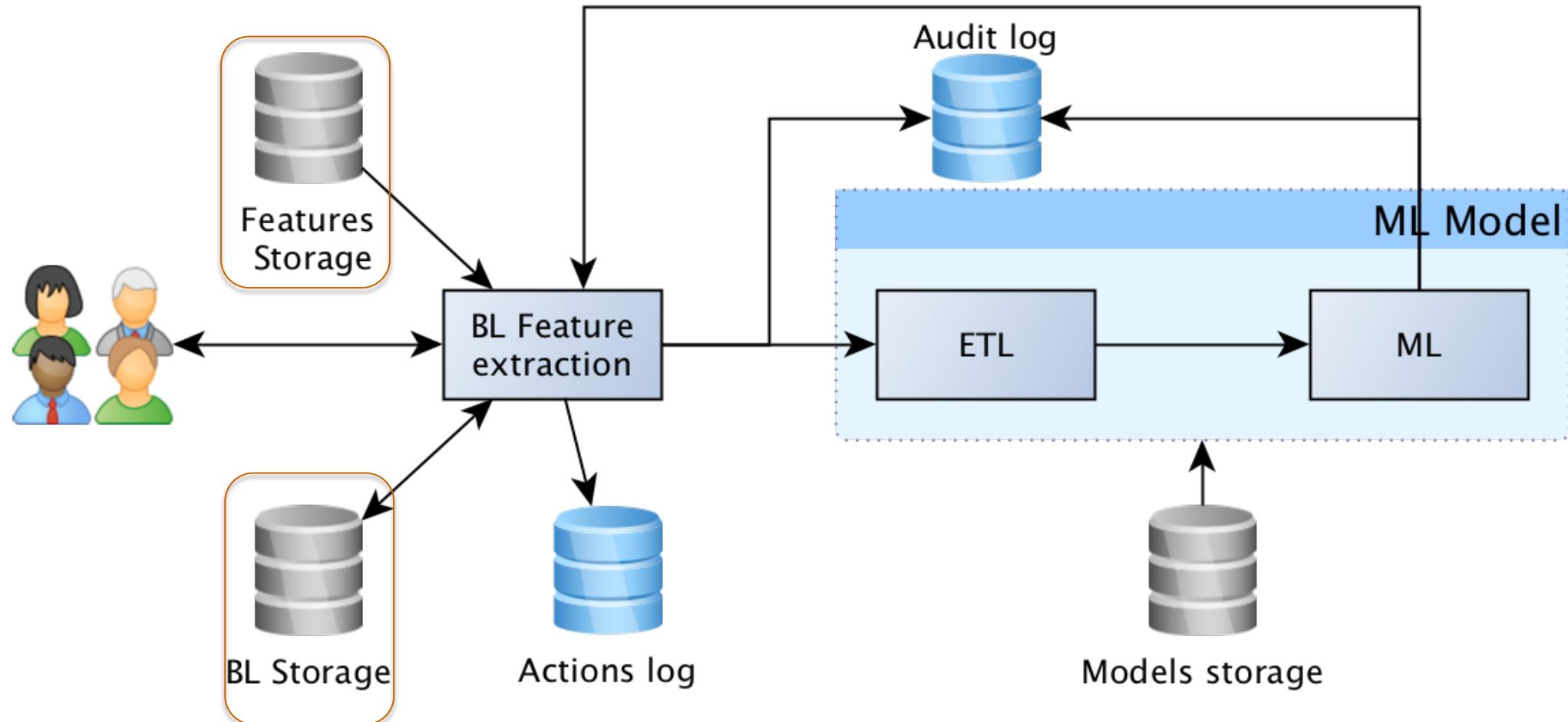
# Serving layer



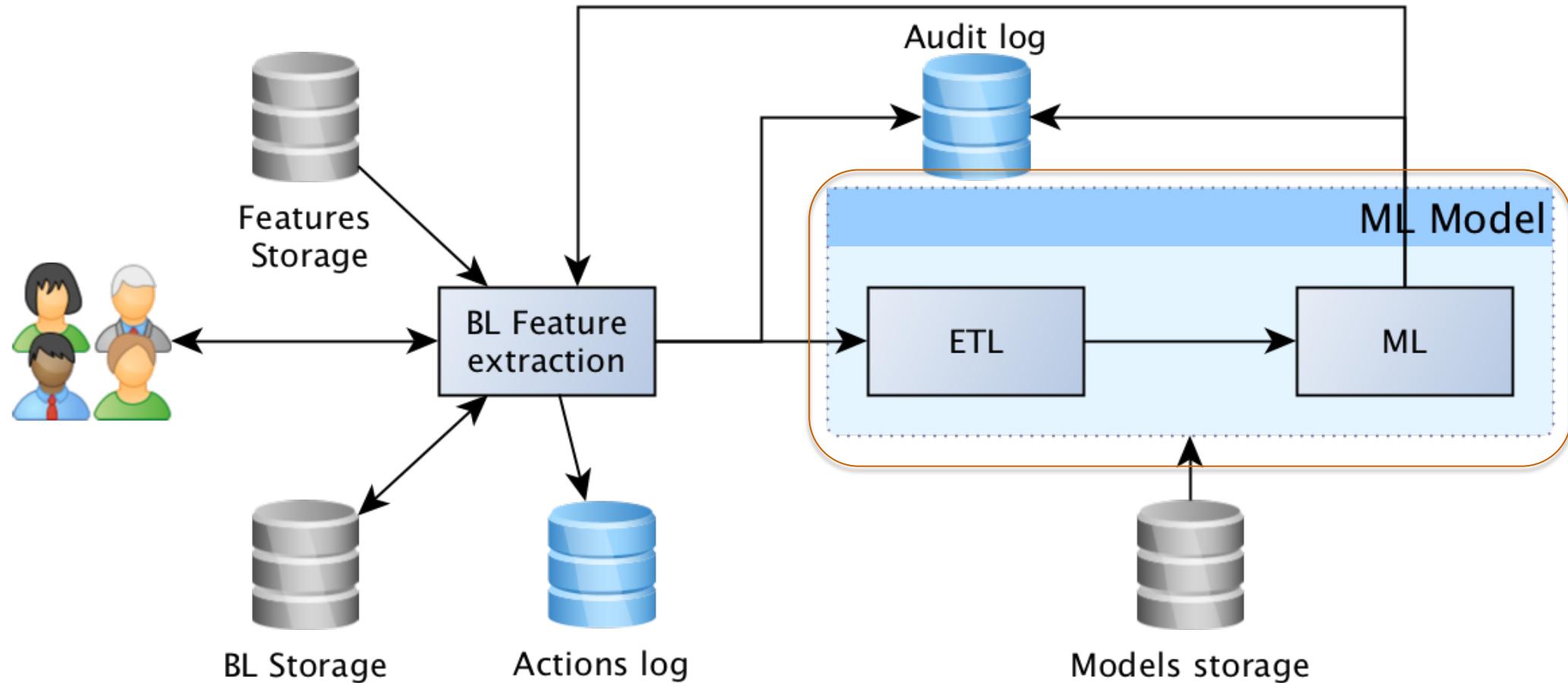
# Serving layer



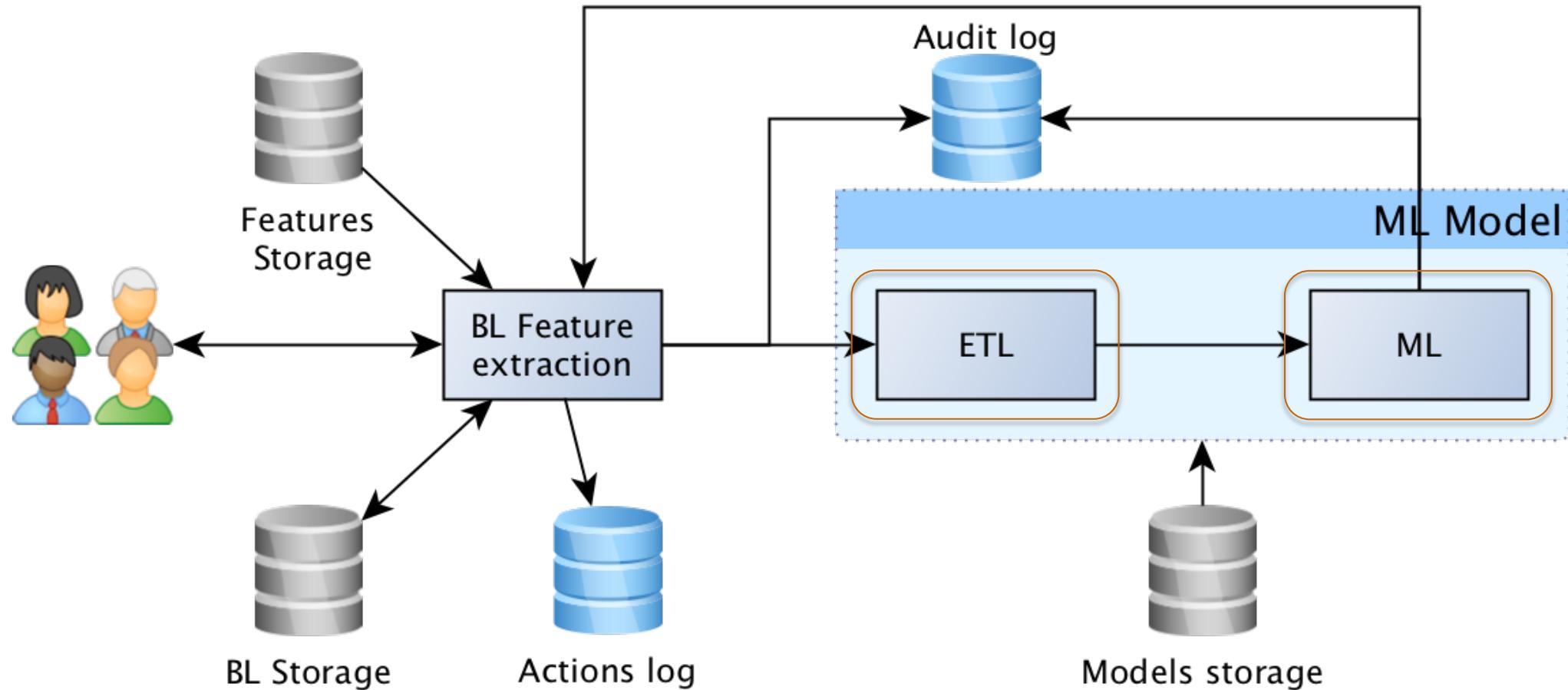
# Serving layer



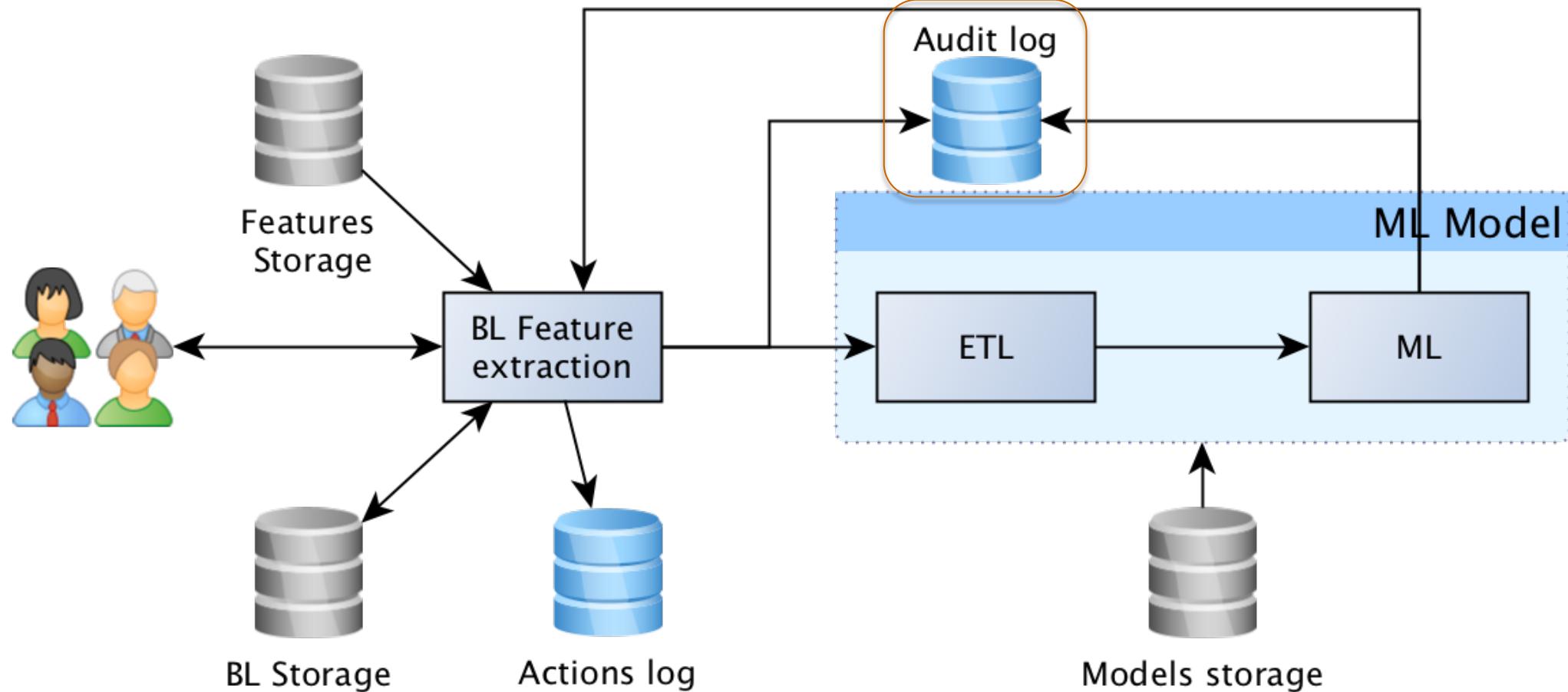
# Serving layer



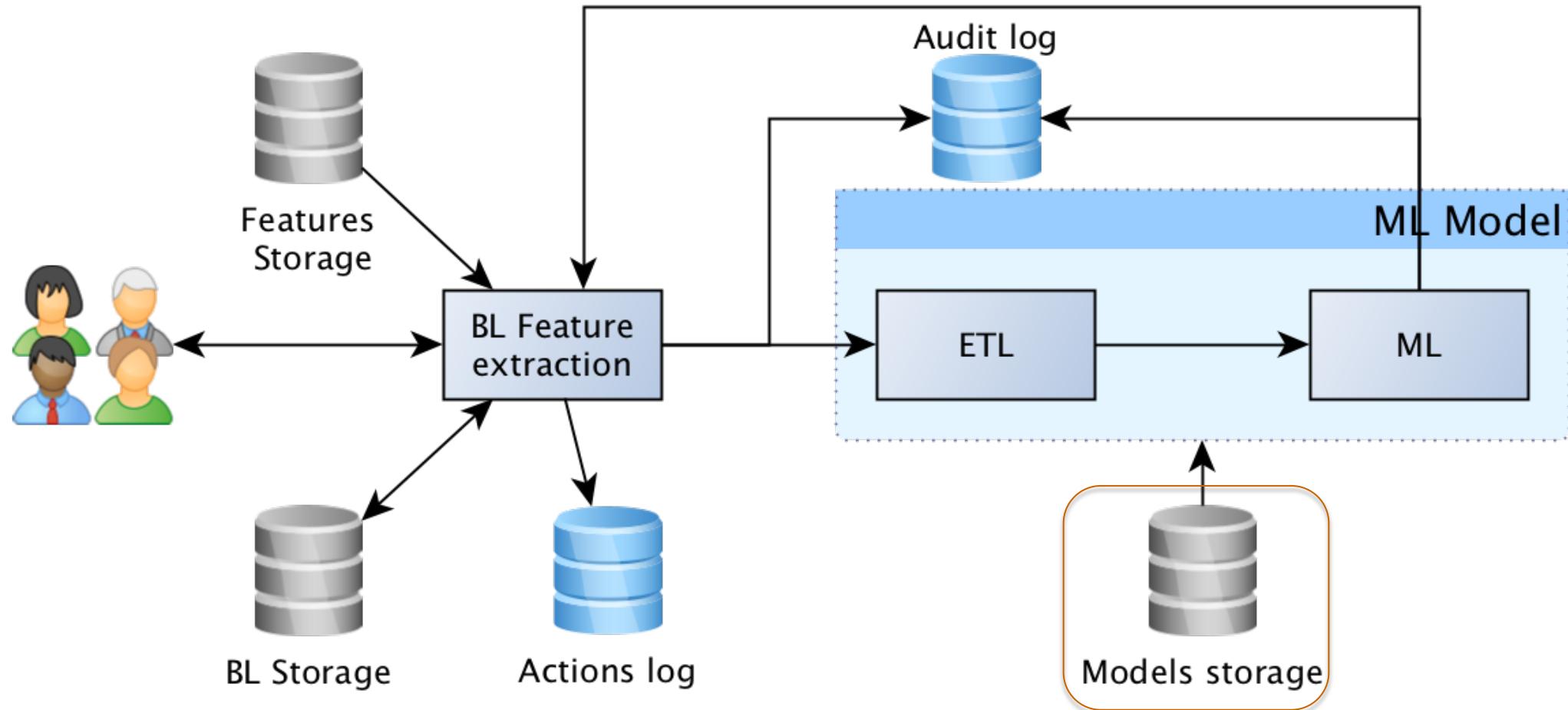
# Serving layer



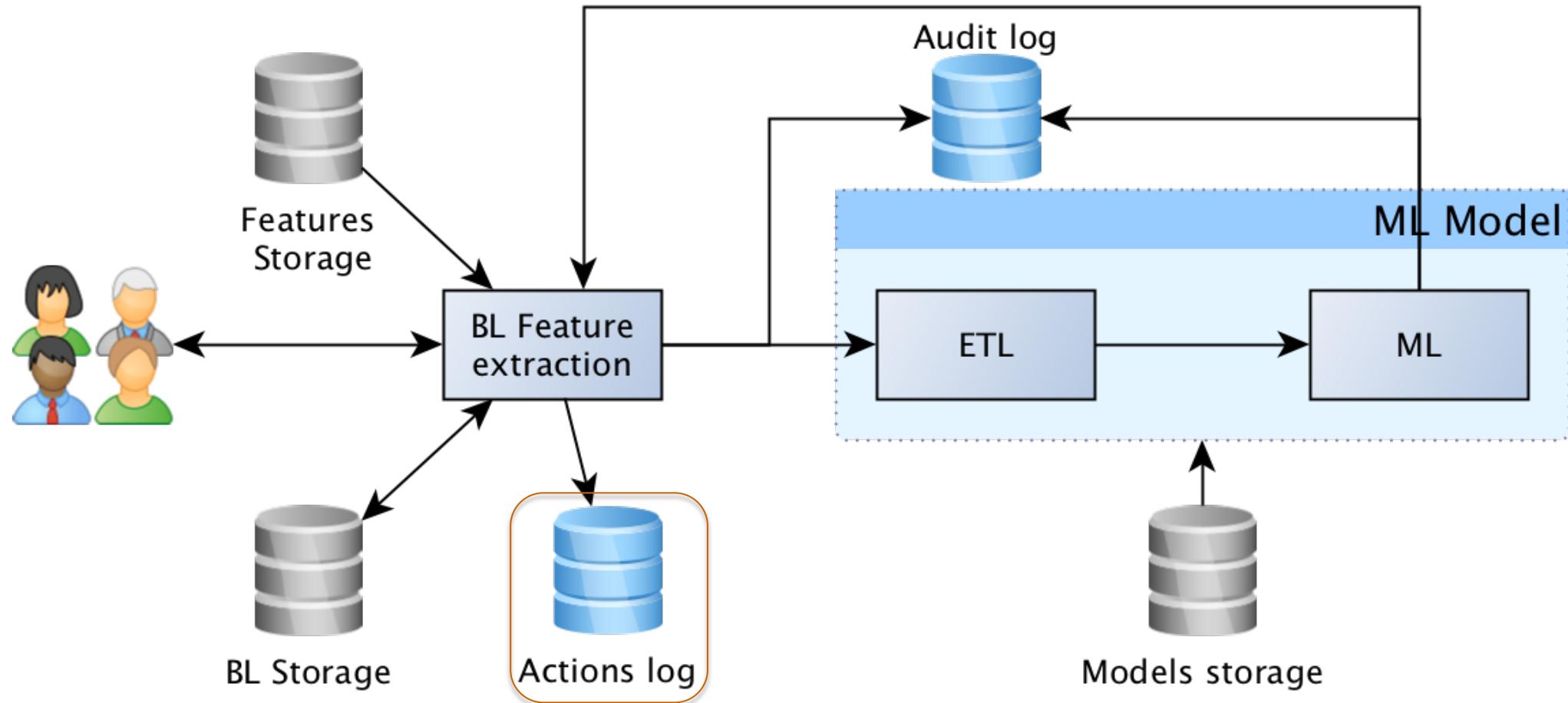
# Serving layer



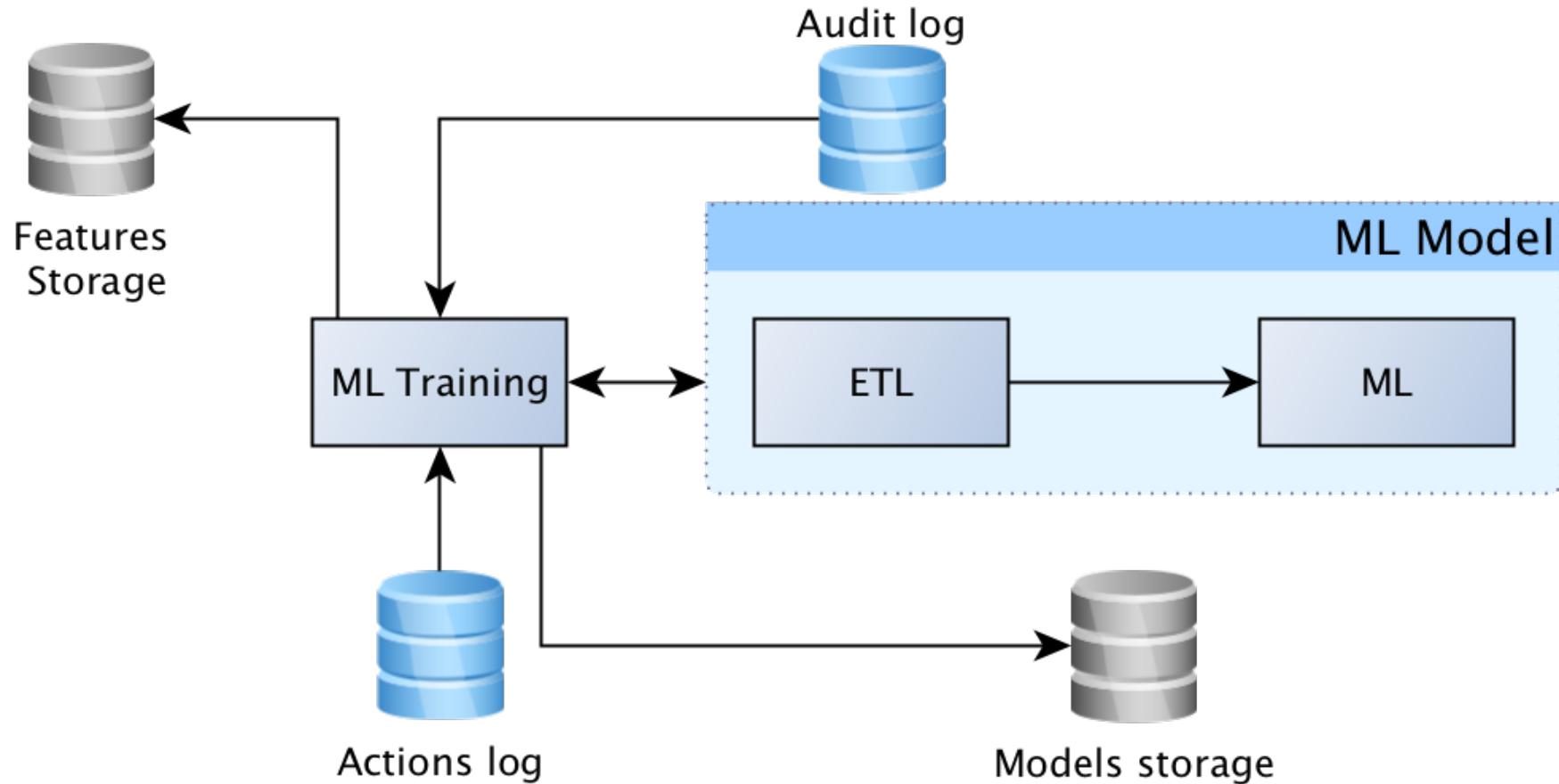
# Serving layer



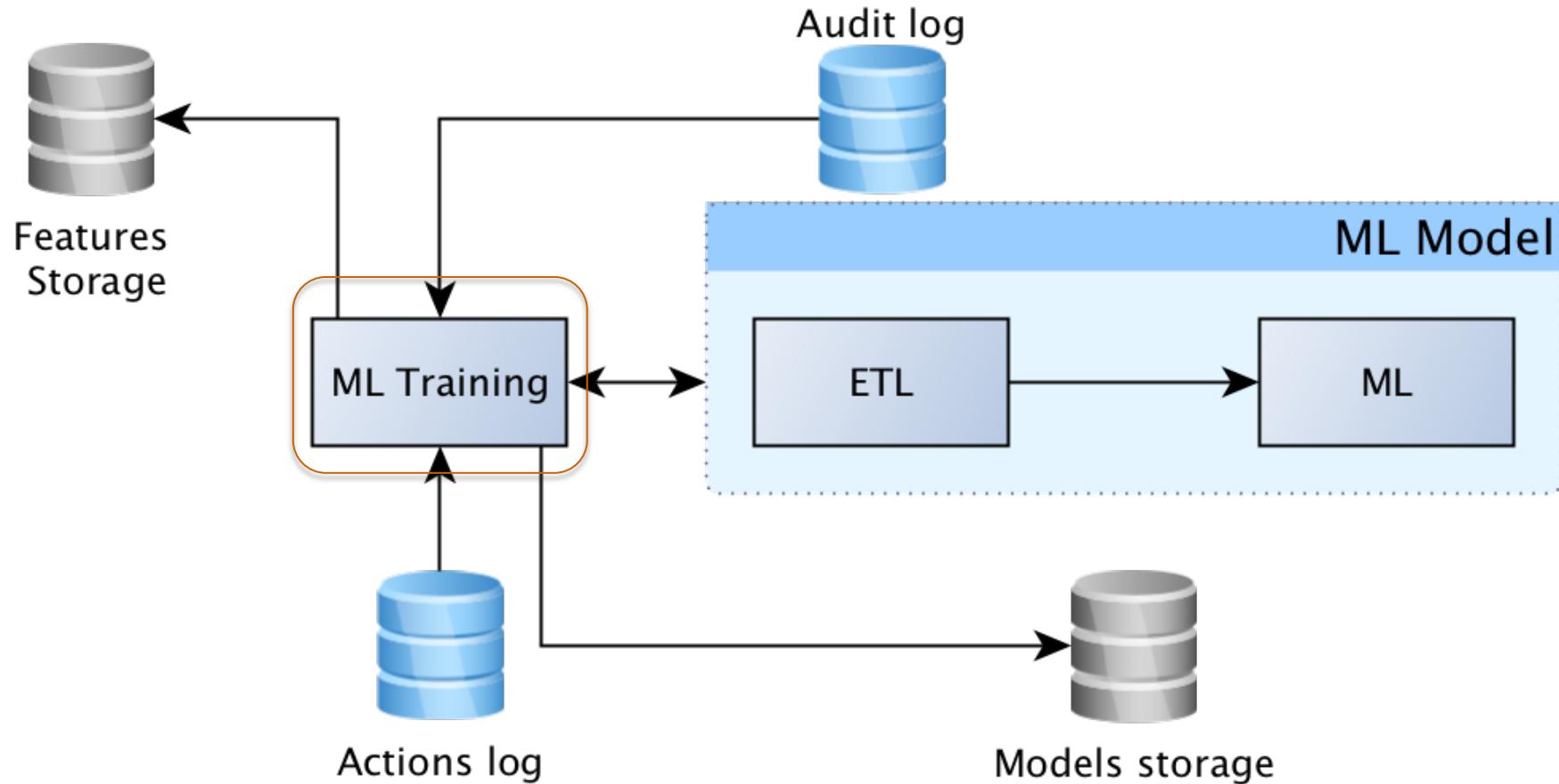
# Serving layer



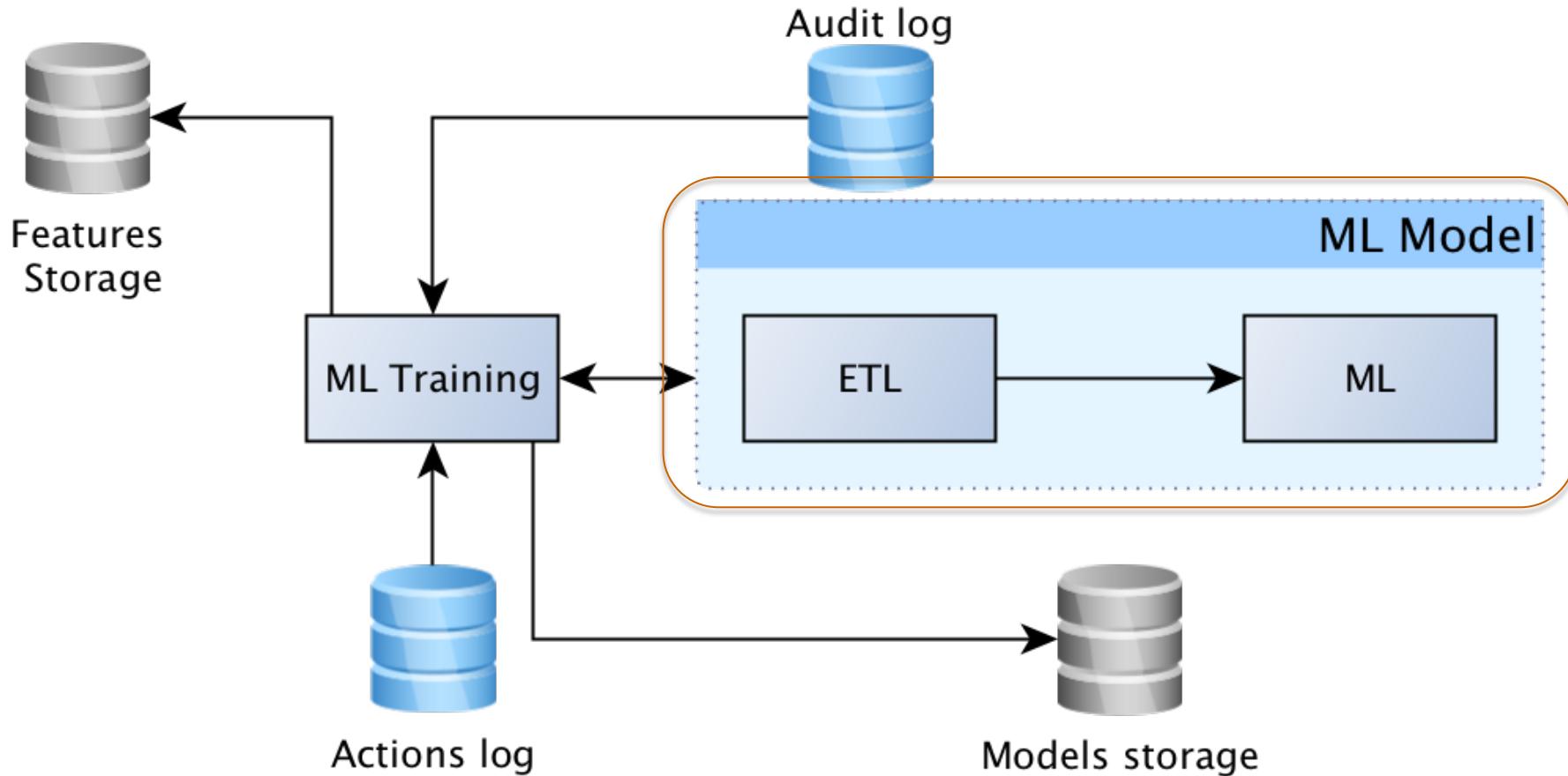
# Training layer



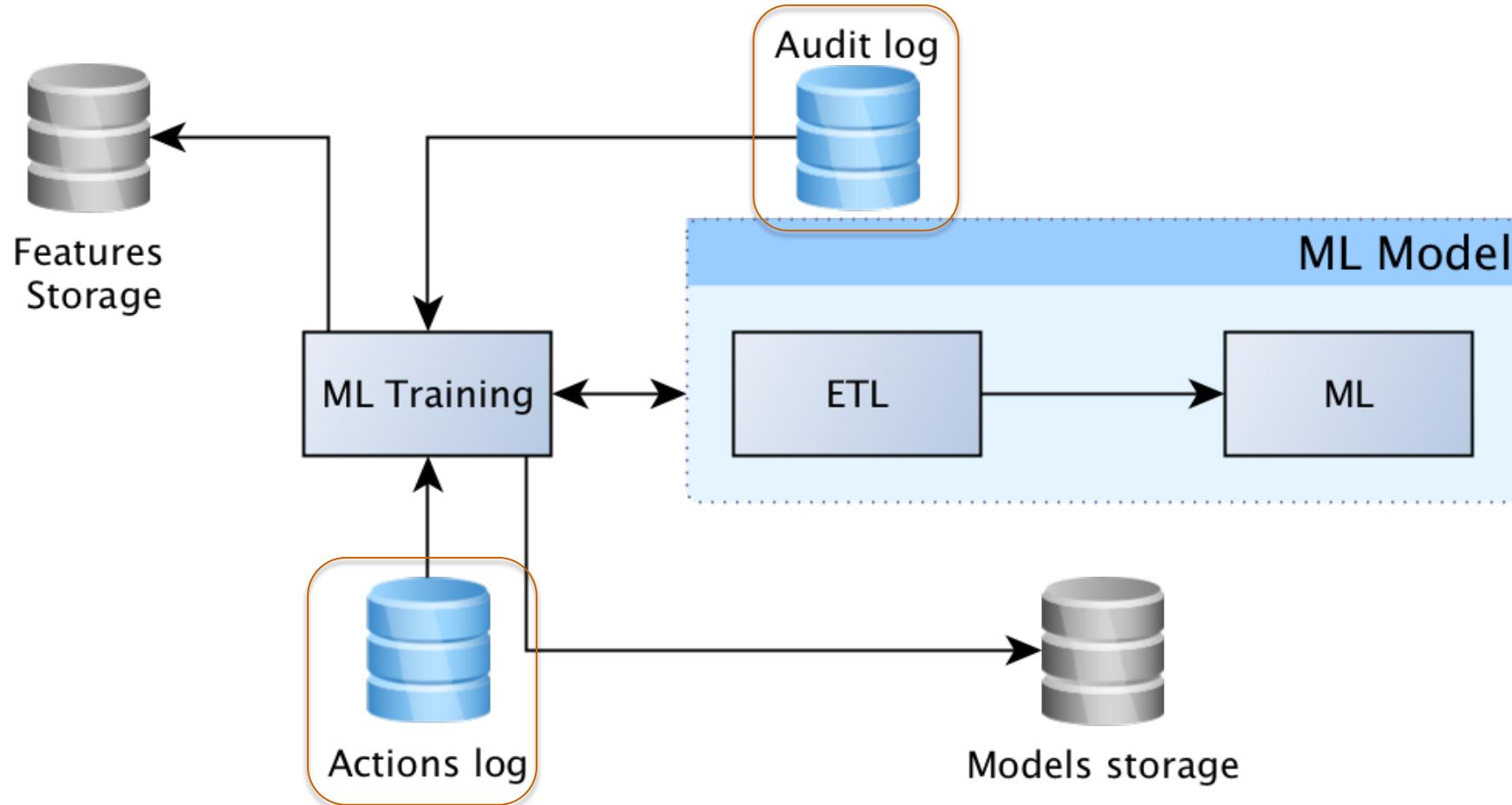
# Training layer



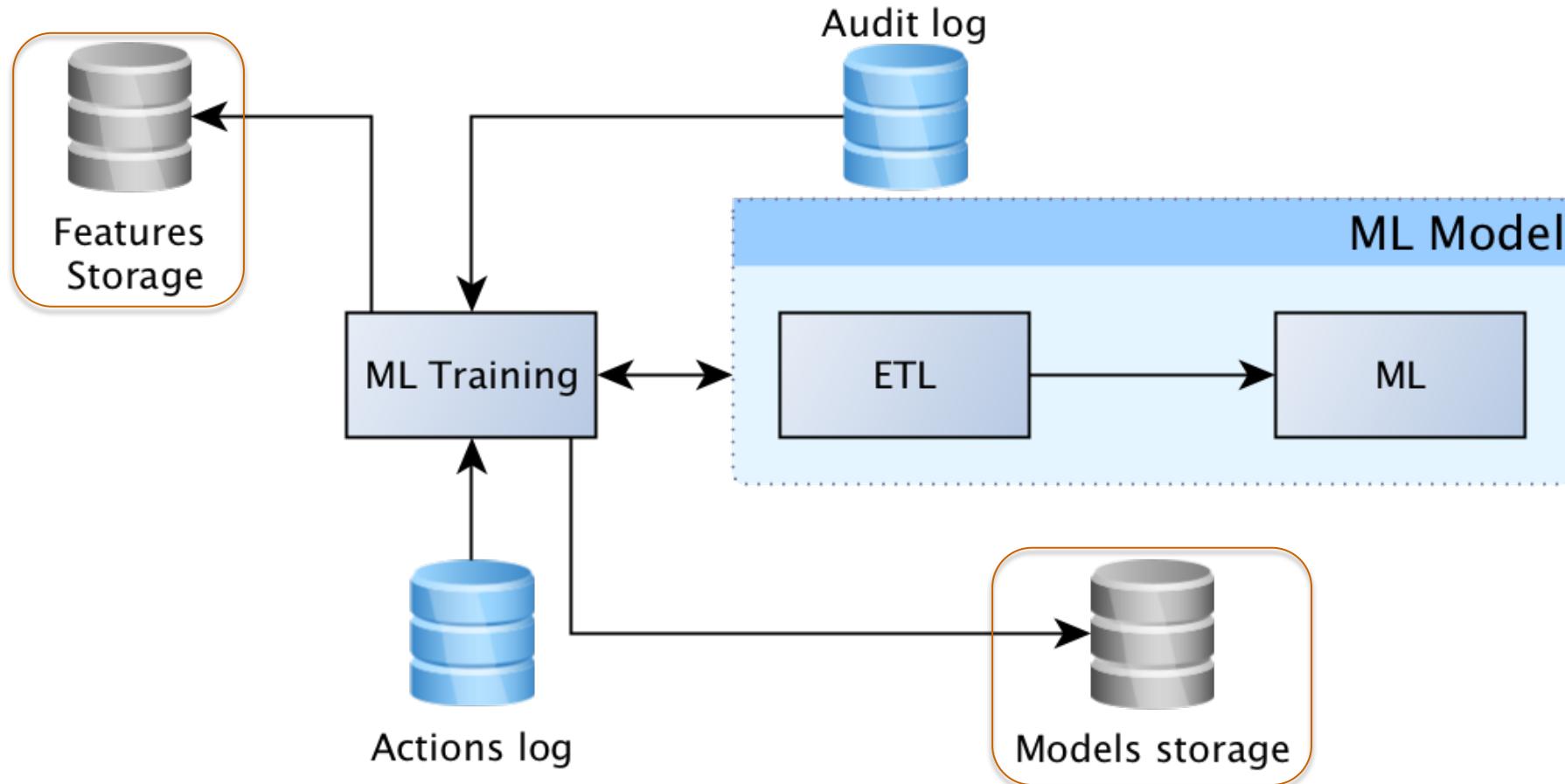
# Training layer



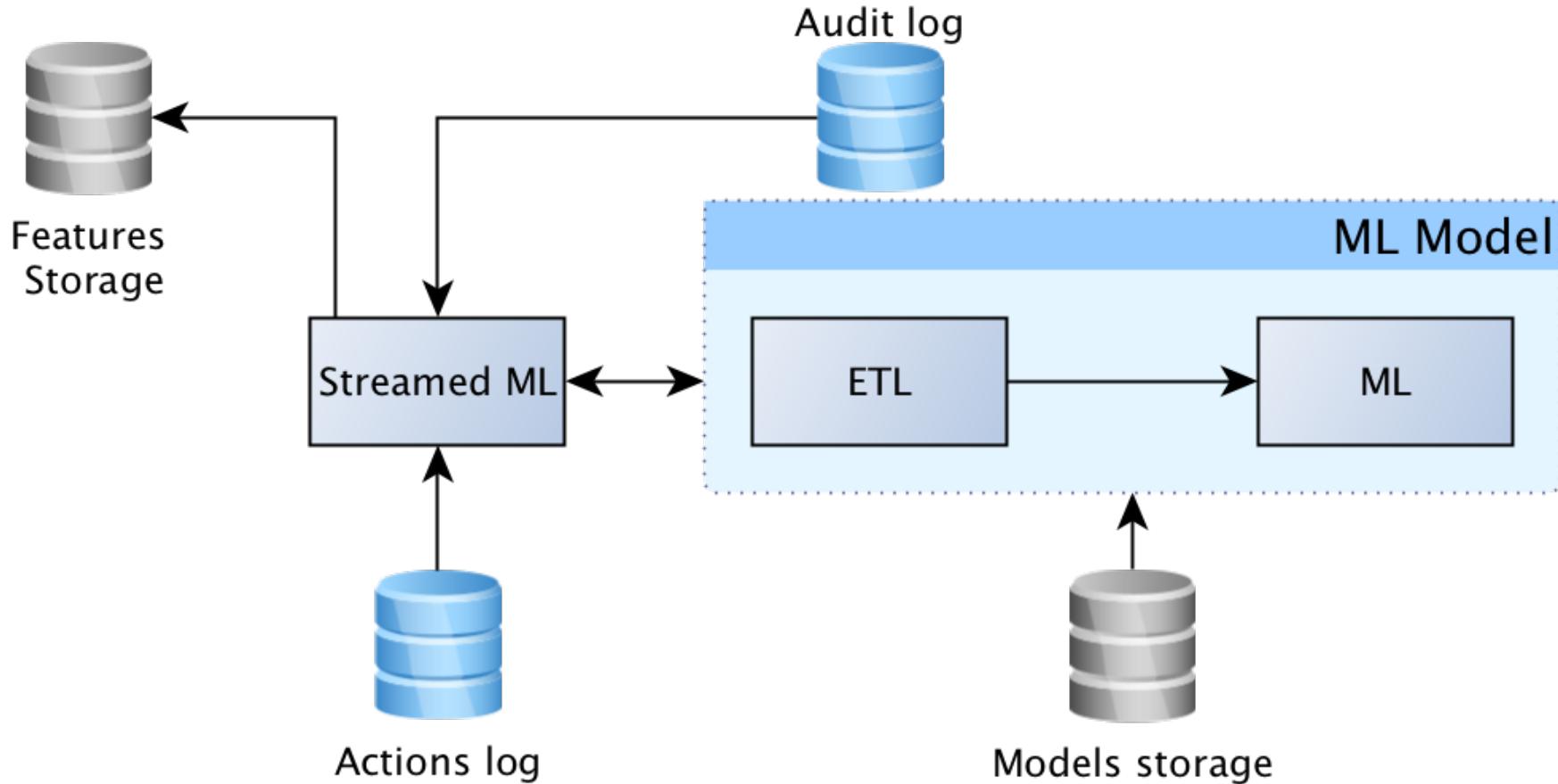
# Training layer



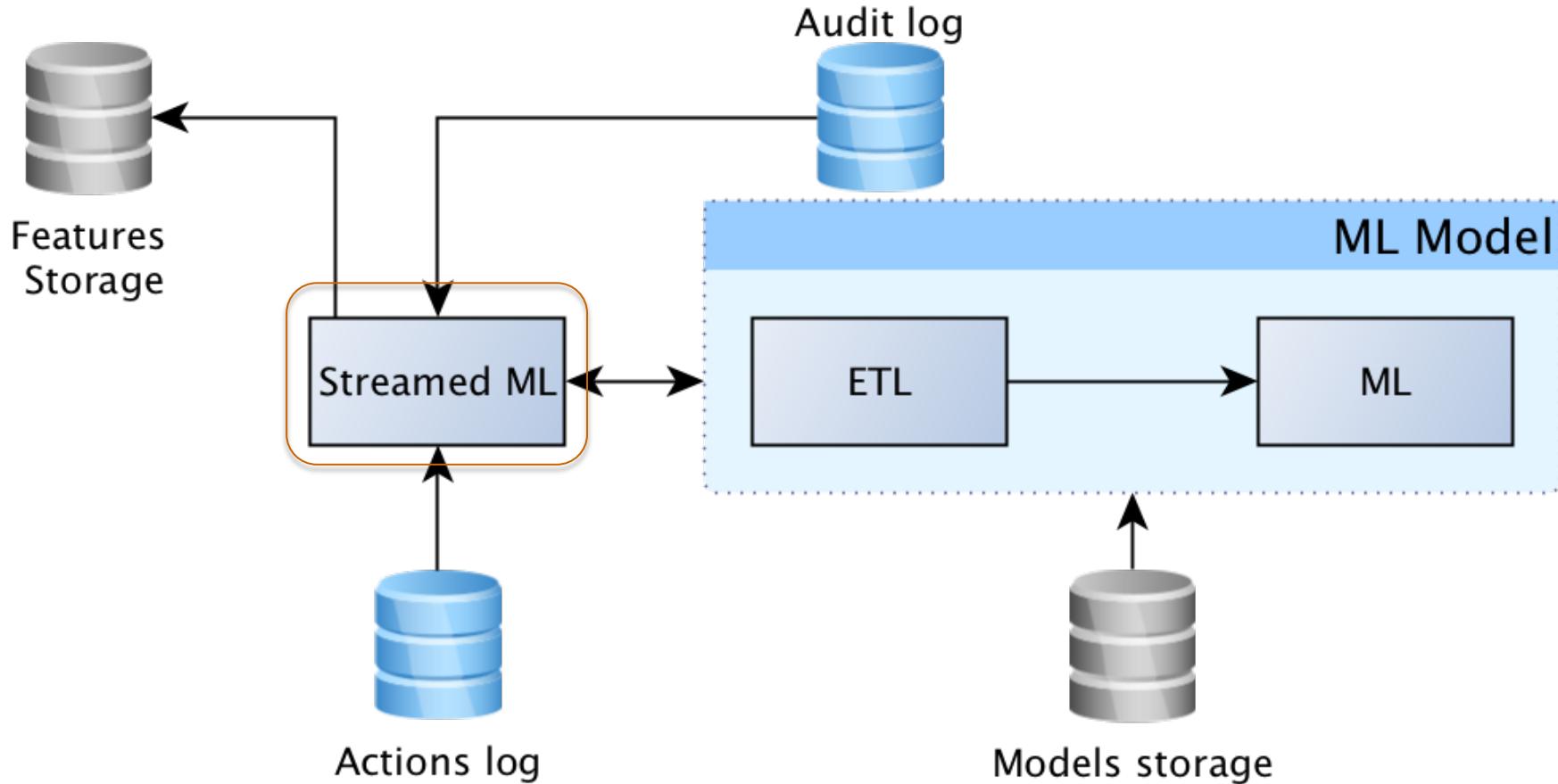
# Training layer



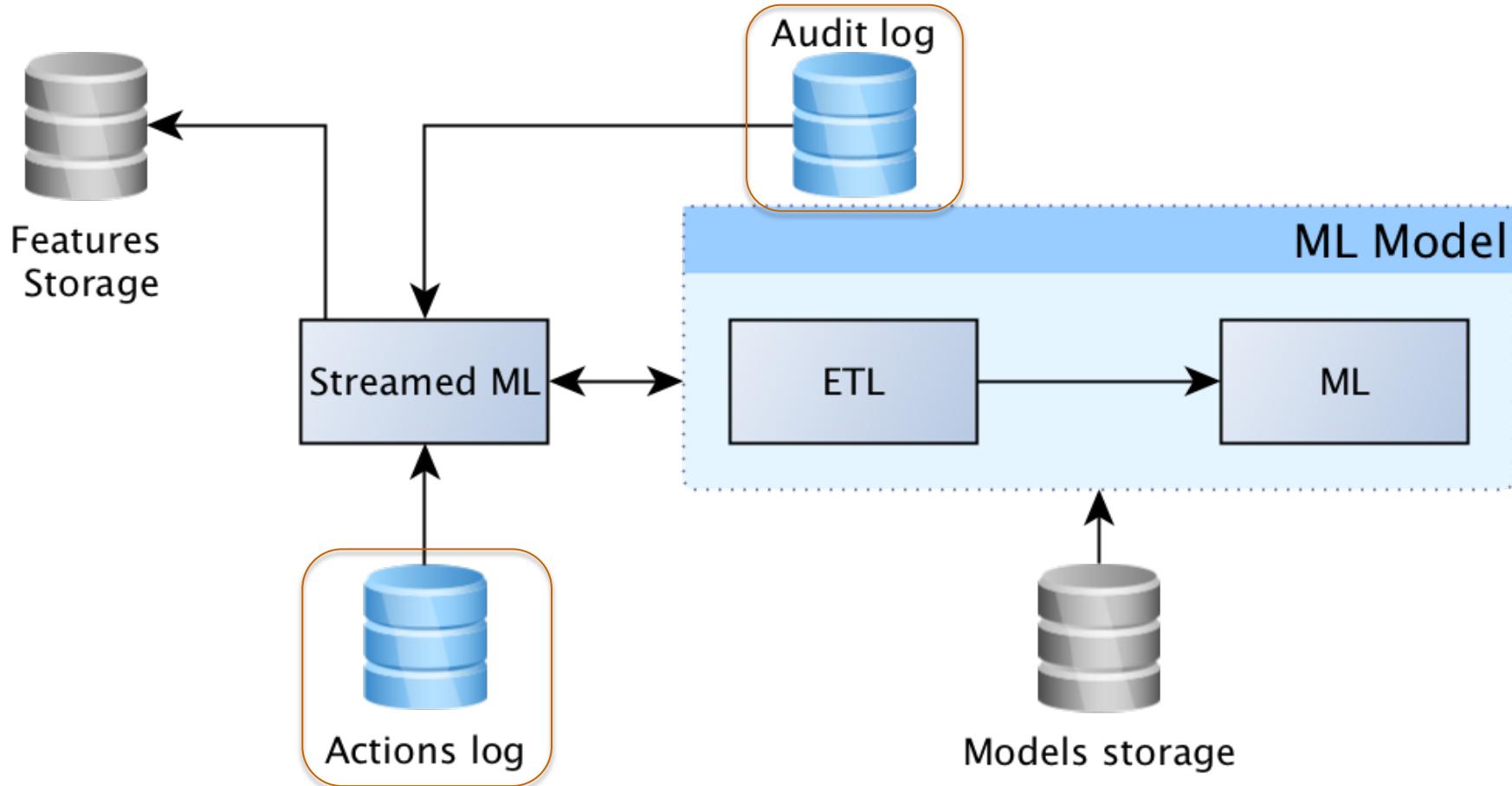
# Streaming adjustments



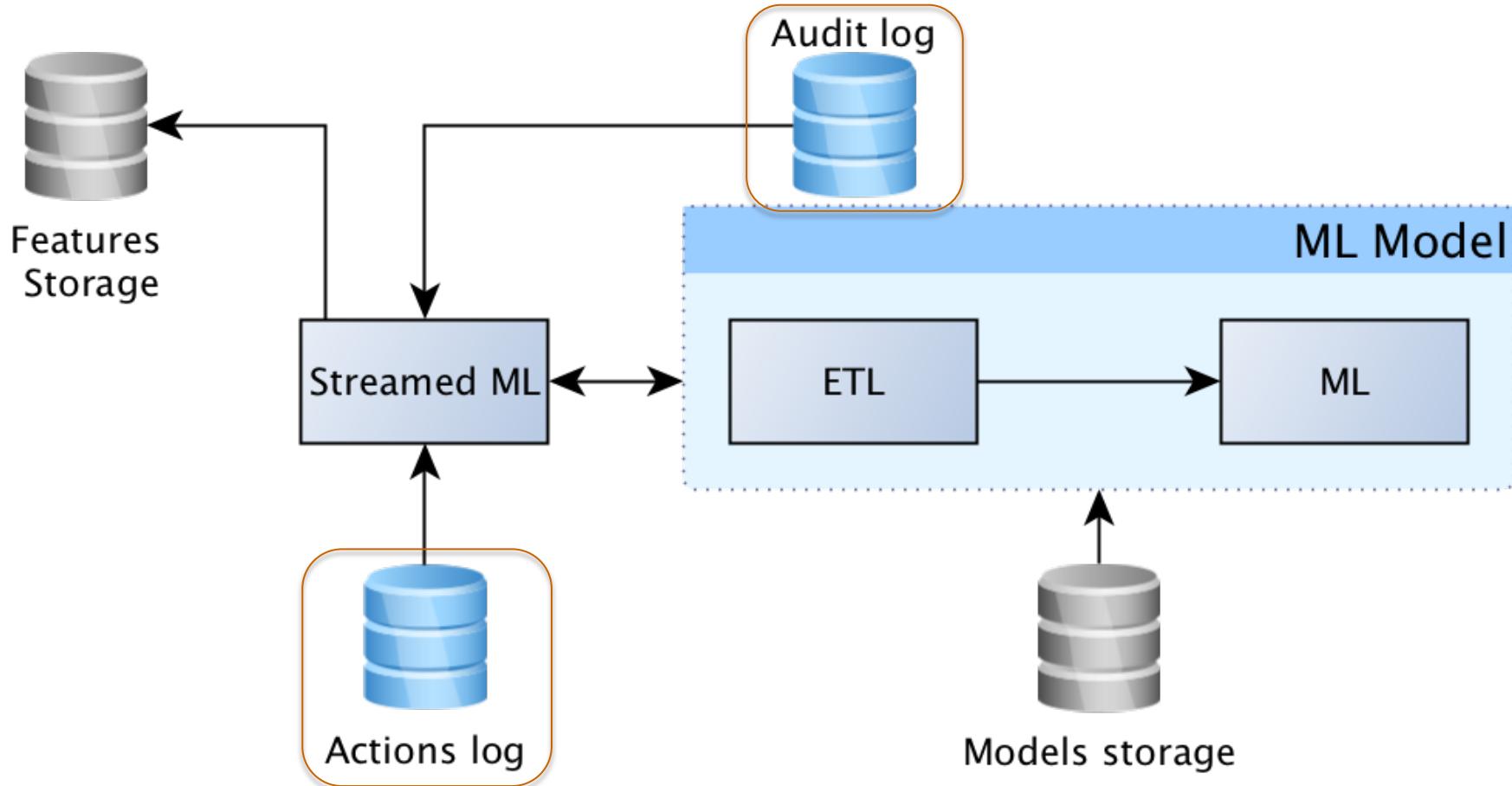
# Streaming adjustments



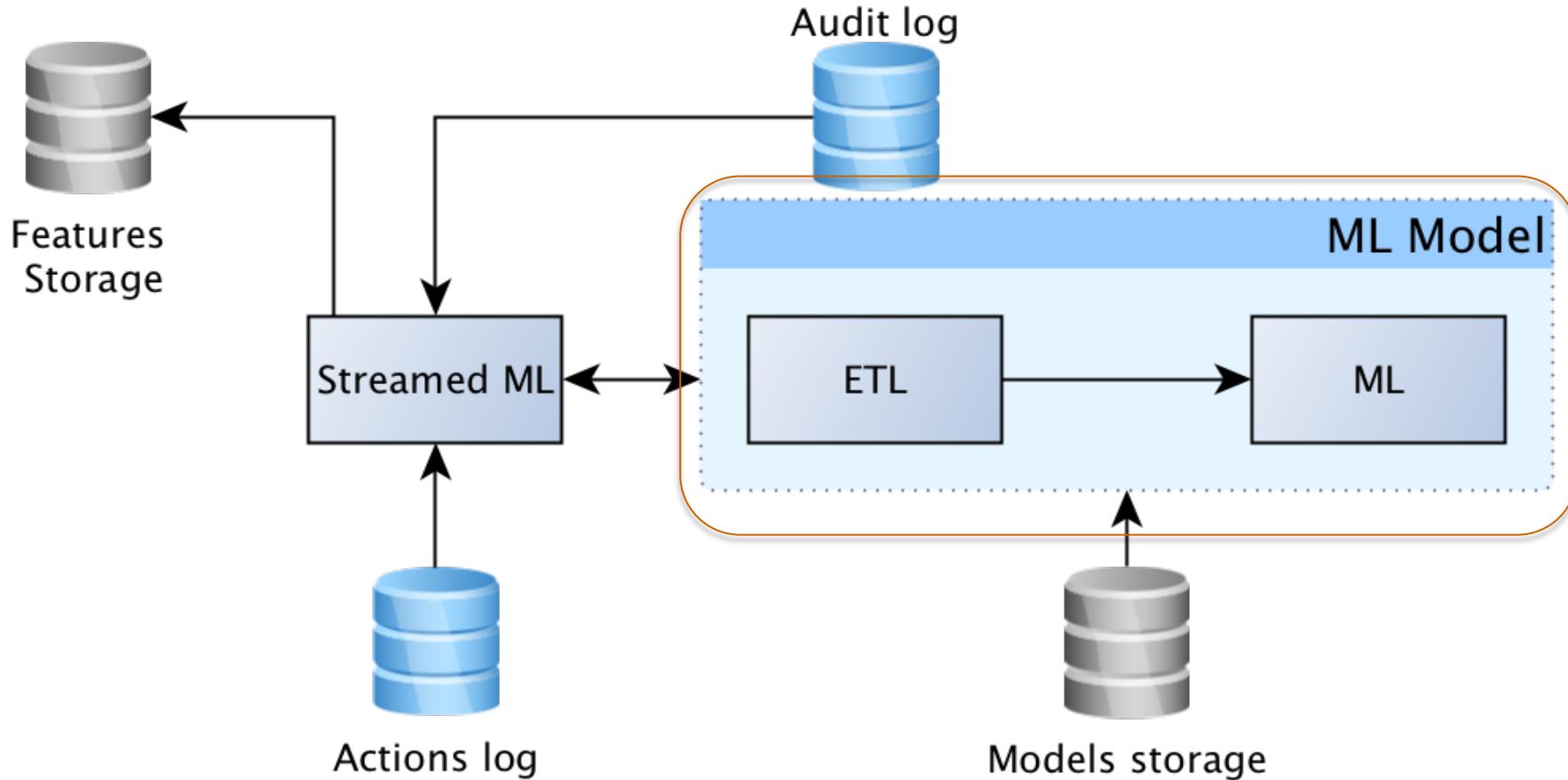
# Streaming adjustments



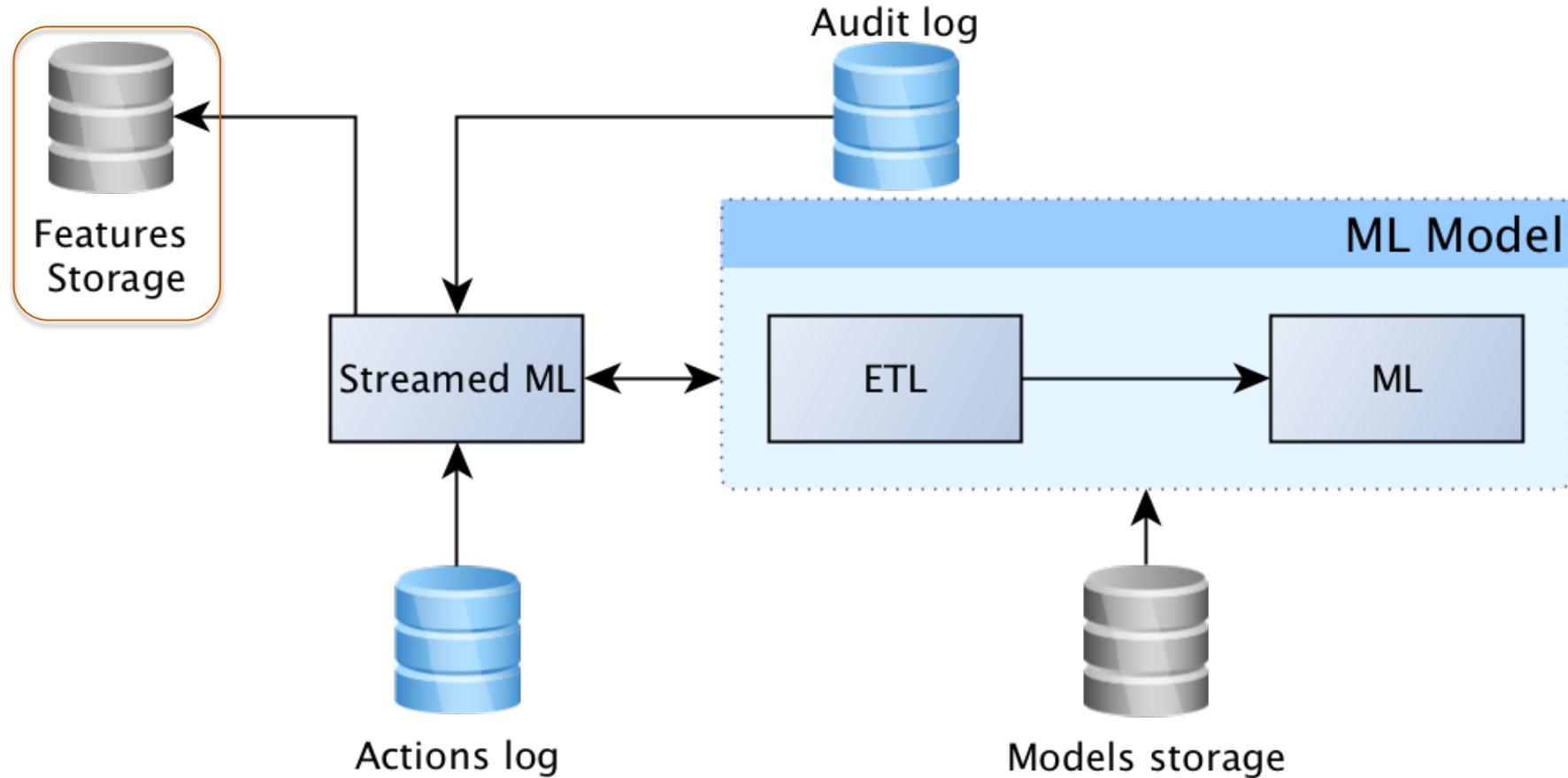
# Streaming adjustments



# Streaming adjustments



# Streaming adjustments



# Платформы потоковой аналитики

- Spark [Structured] Streaming
  - Просто интегрировать, но сложно эксплуатировать
  - Идеально для ad-hoc аналитики
- Apache Samza
  - Больше кода, надежнее результат
- Kafka Streams
- Apache Flink
  - Перспективная новая система

# Примеры из области RecSys

---

# Начало



## Jussi Karlgren:

A paper on the “Digital Bookshelf” was promptly rejected by the 1990 INTERACT reviewers because they held that building a recommender system would interfere with users’ privacy and integrity.

# Начало

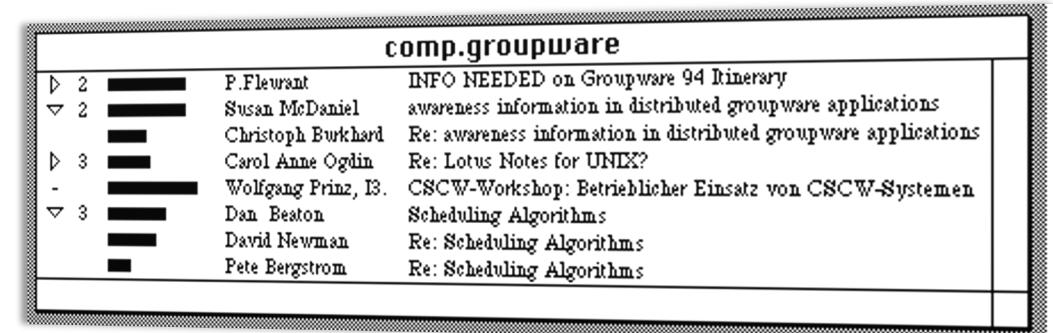
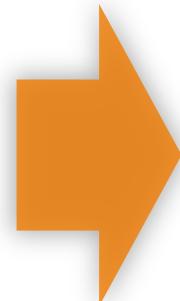
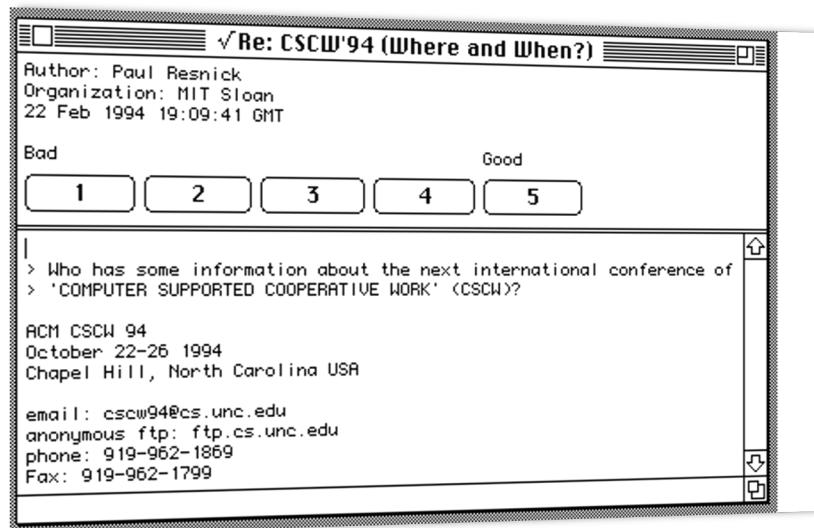


## Jussi Karlgren:

A paper on the “Digital Bookshelf” was promptly rejected by the **1990 INTERACT** reviewers because they held that building a recommender system **would interfere with users’ privacy and integrity.**

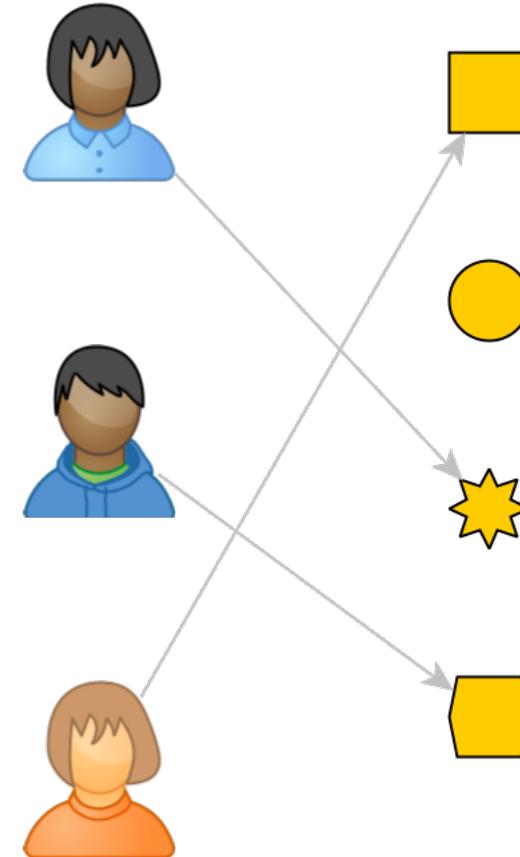
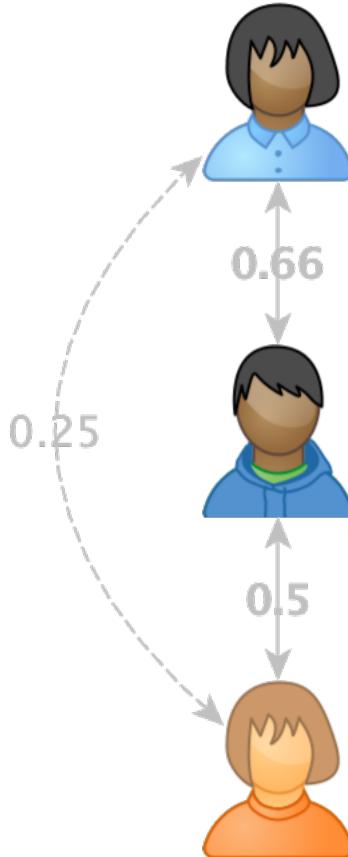
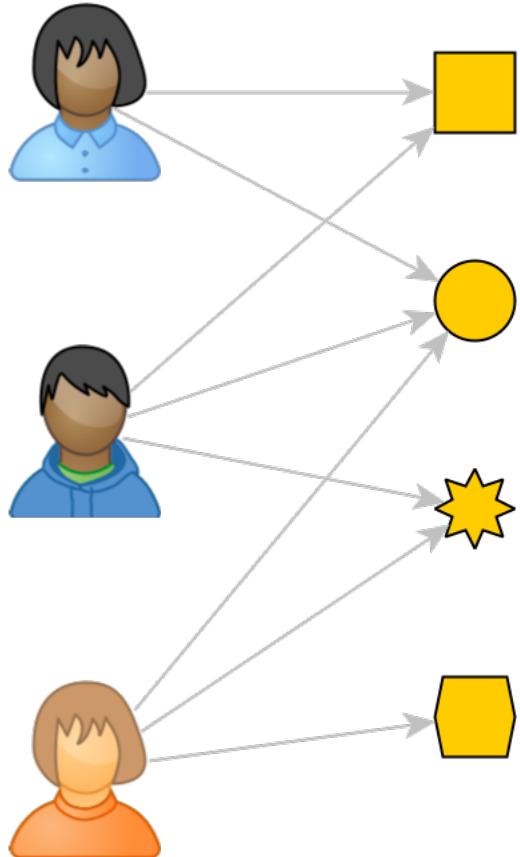
# Наивная юность

Цель: помочь пользователю ориентироваться в  
океане контента

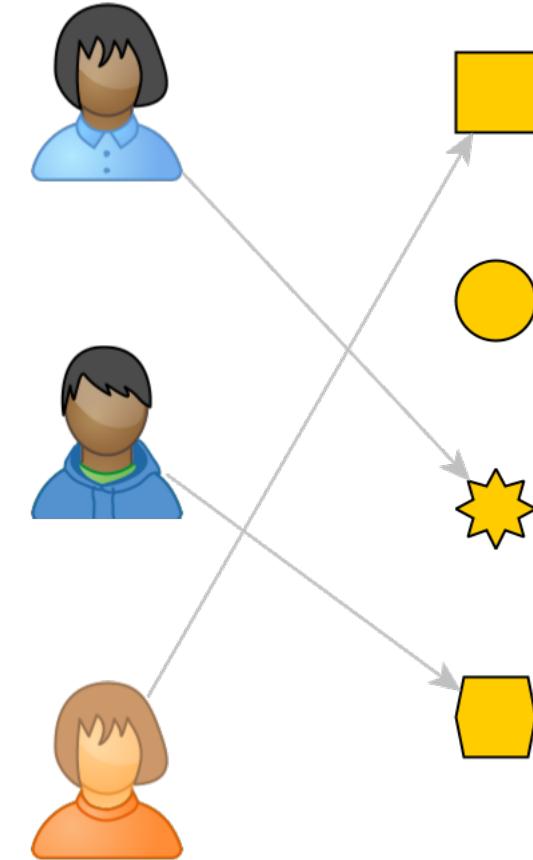
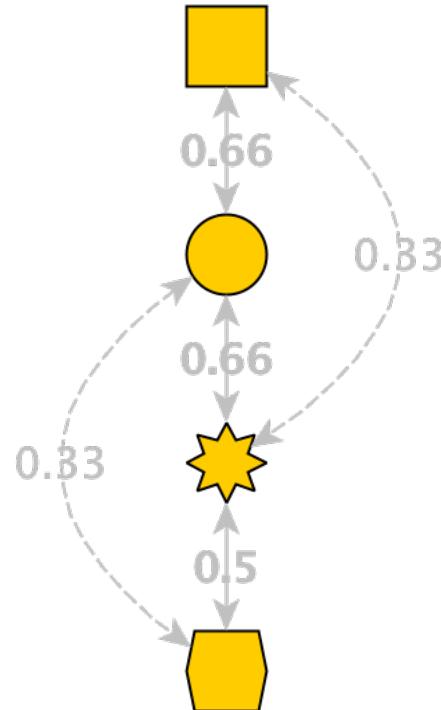
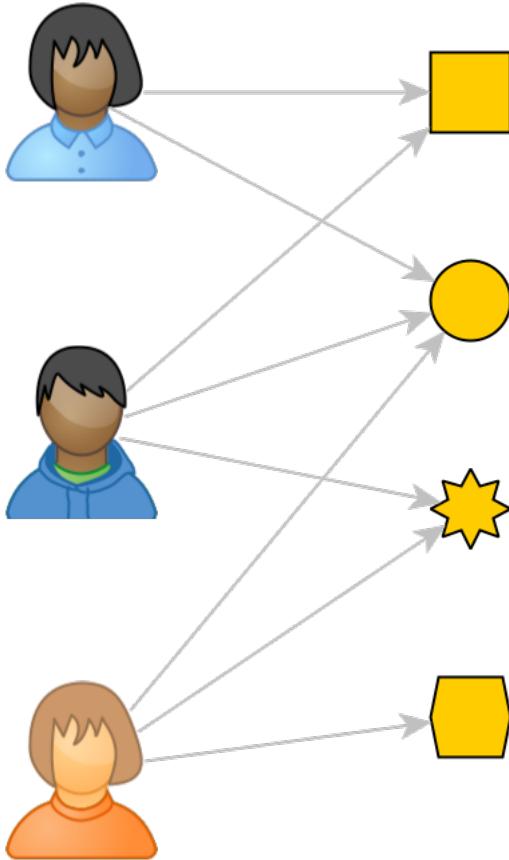


<http://ccs.mit.edu/papers/CCSWP165.html>

Вам понравится то, что нравилось тем, кому нравилось то, что нравилось вам



Вам понравится то, что нравилось тем, кому нравилось то, что нравилось вам

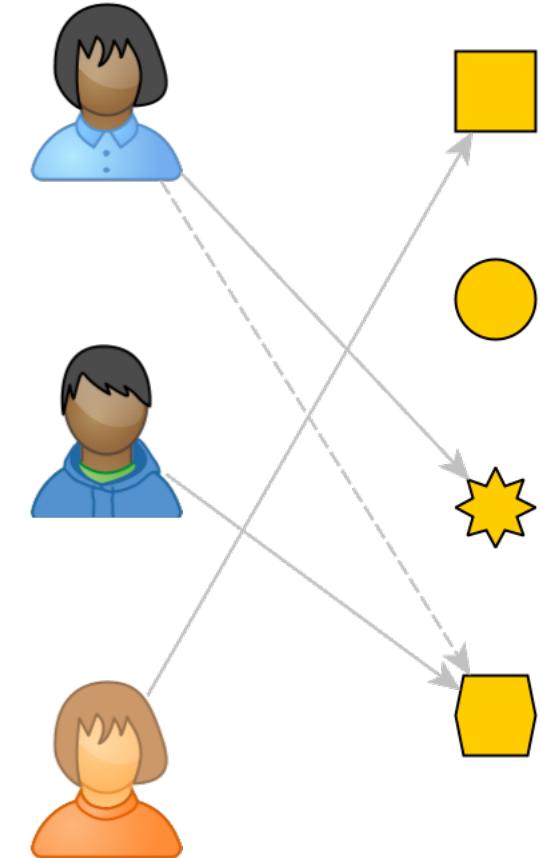
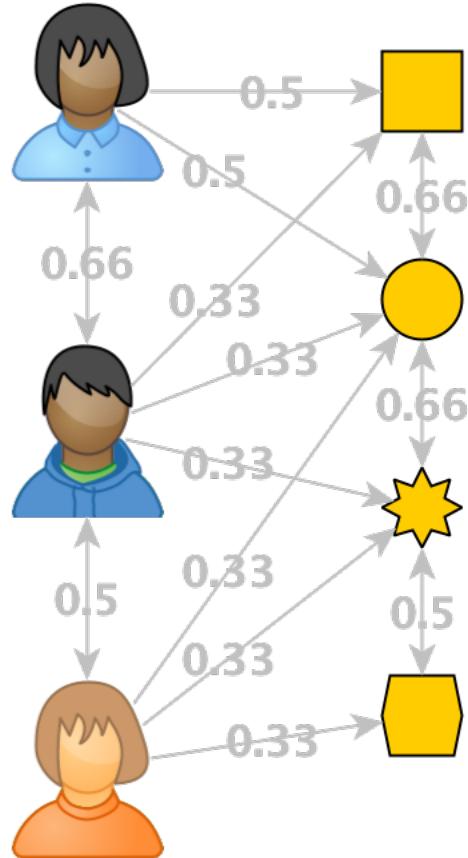
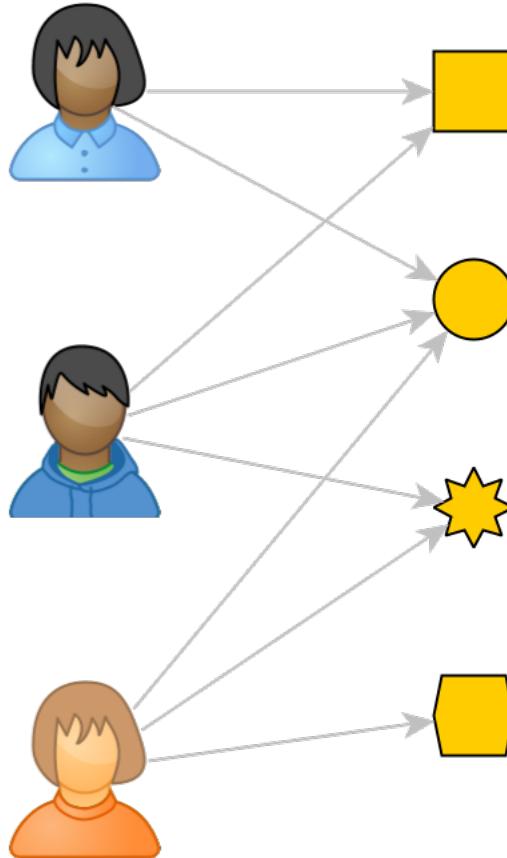


# Item-item vs. User-user

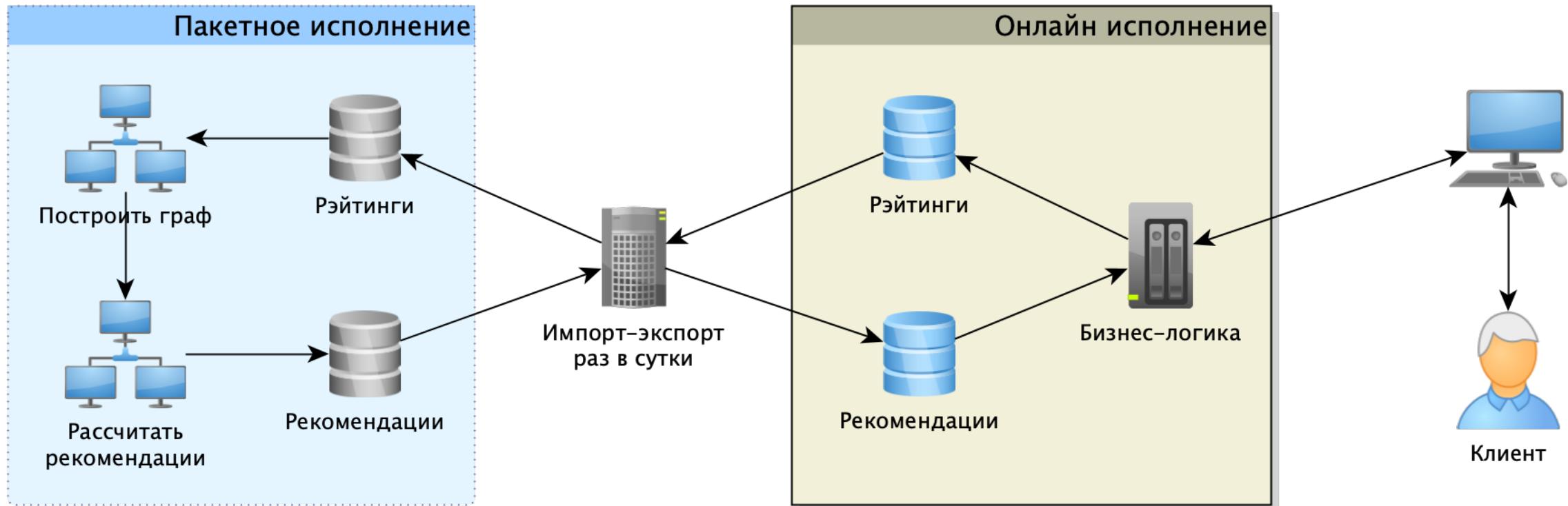
Небольшой  
стабильный каталог  
Меняющиеся  
вкусы клиентов

Большой динамичный  
каталог  
Относительно  
стабильные клиенты

# Item-item vs. User-user – false dilemma



# 1990-е: рекомендации в ПРОМ

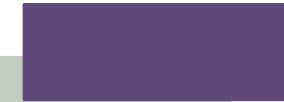


# Построение рекомендаций оффлайн



## Плюсы

- Нет риска "положить ПРОМ"
- Единая среда разработки и исполнения



## Минусы

- Большое количество рекомендаций рассчитано впустую
- Нет возможности сделать большую выдачу
- Долгая реакция на обратную связь пользователя
- Нет учета контекста "в моменте"
- ...

# Подводные камни

- Выкатить «кривые» рекомендации в ПРОМ
  - Мониторинг качества рекомендации перед выкладкой
- Не заметить что рекомендации не обновляются
  - Мониторинг фактов и объемов выгрузок

# 2000-е: Рекомендации в ПРОМ



# Построение рекомендаций онлайн



## Плюсы

- Возможность учета обратной связи от пользователя онлайн
- Возможность учета оперативного контекста



## Минусы

- Возможность уложить ПРОМ

# Простые способы уложить ПРОМ

- Пустить чрезмерную нагрузку на ПРОМ базы данных (рейтинги, метаданные и т.д.)
  - Разделить базы данных
  - Прокидывать данные через бизнес-логику
- Долго строить рекомендации
  - Оптимизировать код и структуры данных ☺

# Задача 1

## Дано

- Граф вкусов
  - 1 миллион вершин
  - 100 миллионов связей
- Используем структуру данных
  - `HashMap<Integer,HashMap<Integer,Float>>`

## Найти

- Сколько памяти нужно серверу чтобы работать с таким графом?

# Задача 1

## Дано

- Граф вкусов
  - 1 миллион вершин
  - 100 миллионов связей
- Используем структуру данных
  - `HashMap<Integer,HashMap<Integer,Float>>`

## Найти

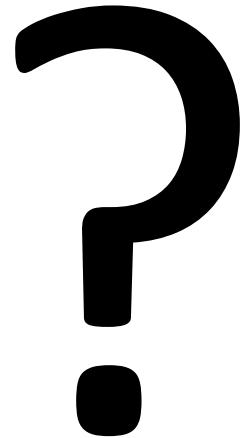
- Сколько памяти нужно серверу чтобы работать с таким графом?

# Задача 1: Подсказка

## Полезная информация

- 100 млн дуг \* (4 + 4 + 4)  
= 1 200 млн байт  
= 1.2 ГБ

## Накладные расходы на дугу



# Задача 1

## Дано

- Граф вкусов
  - 1 миллион вершин
  - 100 миллионов связей
- Используем структуру данных
  - `HashMap<Integer,HashMap<Integer,Float>>`

## Найти

- Сколько памяти нужно серверу чтобы работать с таким графом?
  1. 4 Gb
  2. 8 Gb
  3. 16 Gb
  4. 32 Gb
  5. 64 Gb

# Задача 1

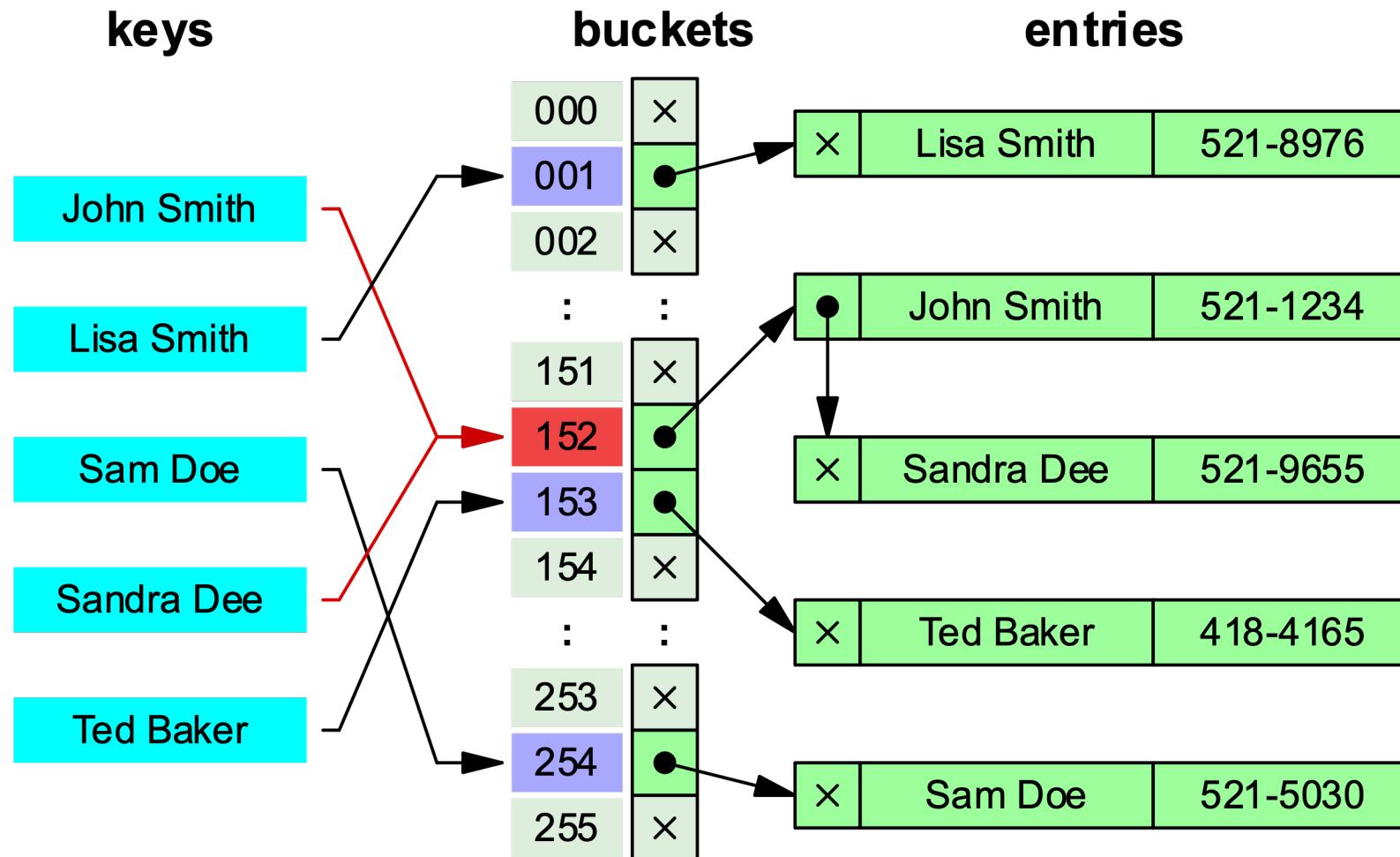
## Дано

- Граф вкусов
  - 1 миллион вершин
  - 100 миллионов связей
- Используем структуру данных
  - `HashMap<Integer,HashMap<Integer,Float>>`

## Найти

- Сколько памяти нужно серверу чтобы работать с таким графом?
  1. 4 Gb
  2. 8 Gb
  3. 16 Gb
  4. 32 Gb
  5. **64 Gb**

# Хэш-таблица: устройство



# Накладные расходы

- Ссылка на запись +8 байт
- Заголовок объекта-записи +16 байт
- Ссылка на следующий элемент +8 байт
- Ссылка на ключ +8 байт
- Заголовок объекта-коробки ключа +16 байт
- Ссылка на объект-значение +8 байт
- Заголовок объекта-коробки ключа +16 байт

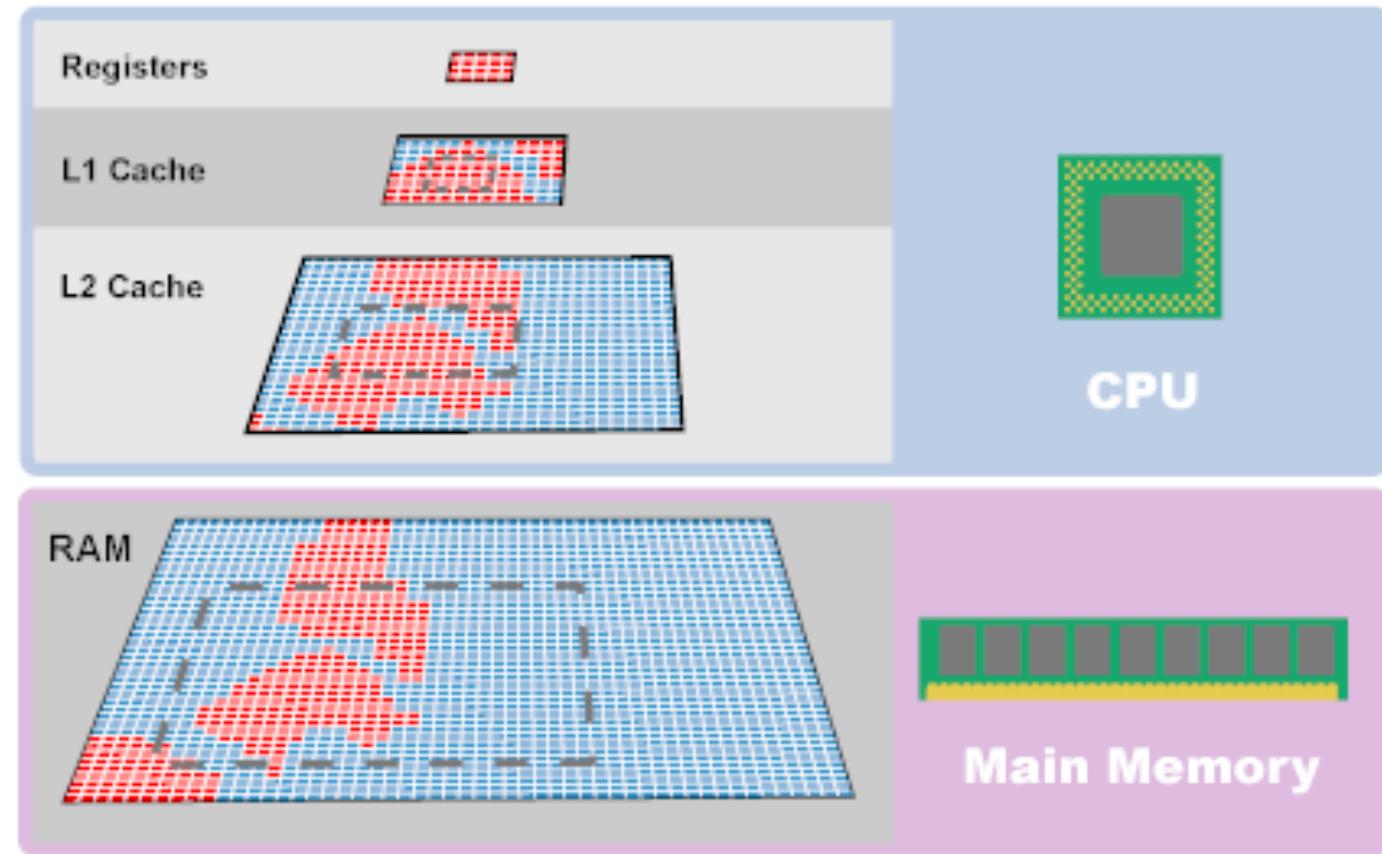
# Накладные расходы

+80 байт  
накладных  
расходов на  
дугу!

- Ссылка на запись +8 байт
- Заголовок объекта-записи +16 байт
- Ссылка на следующий элемент +8 байт
- Ссылка на ключ +8 байт
- Заголовок объекта-коробки ключа +16 байт
- Ссылка на объект-значение +8 байт
- Заголовок объекта-коробки ключа +16 байт

# Локальность данных - о вреде ссылок

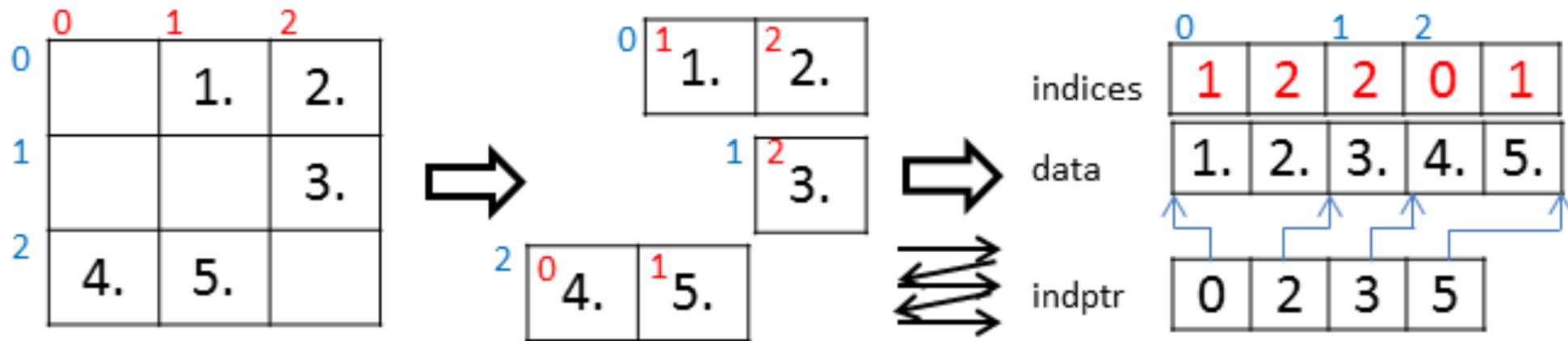
| Source   | Latency   |
|----------|-----------|
| L1 Cache | 1-2 ns    |
| L2 Cache | 3-5 ns    |
| L3 Cache | 10-40 ns  |
| DRAM     | 60-100 ns |



# Выбираем альтернативную структуру данных

- Без ссылок
- Без боксинга
- С быстрым доступом

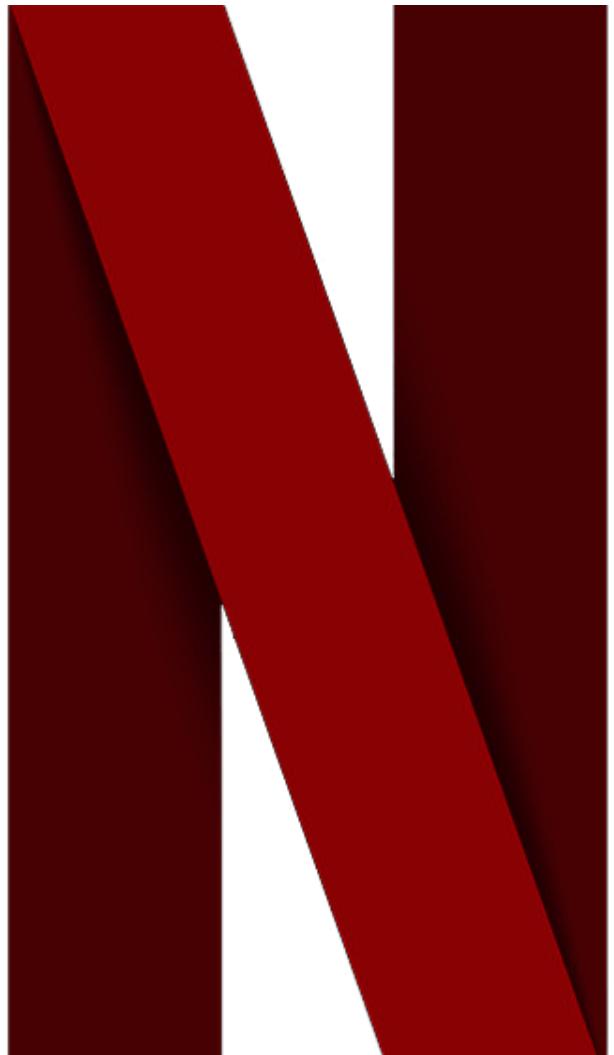
CSR Матрица



# Key take aways

- Вычислять МЛ в онлайне можно
- Выбор структуры данных для представления модели может сыграть решающую роль
- При выборе структуры данных стоит избегать ссылок и боксинга

# Эпоха Netflix prize



# Эпоха Netflix prize

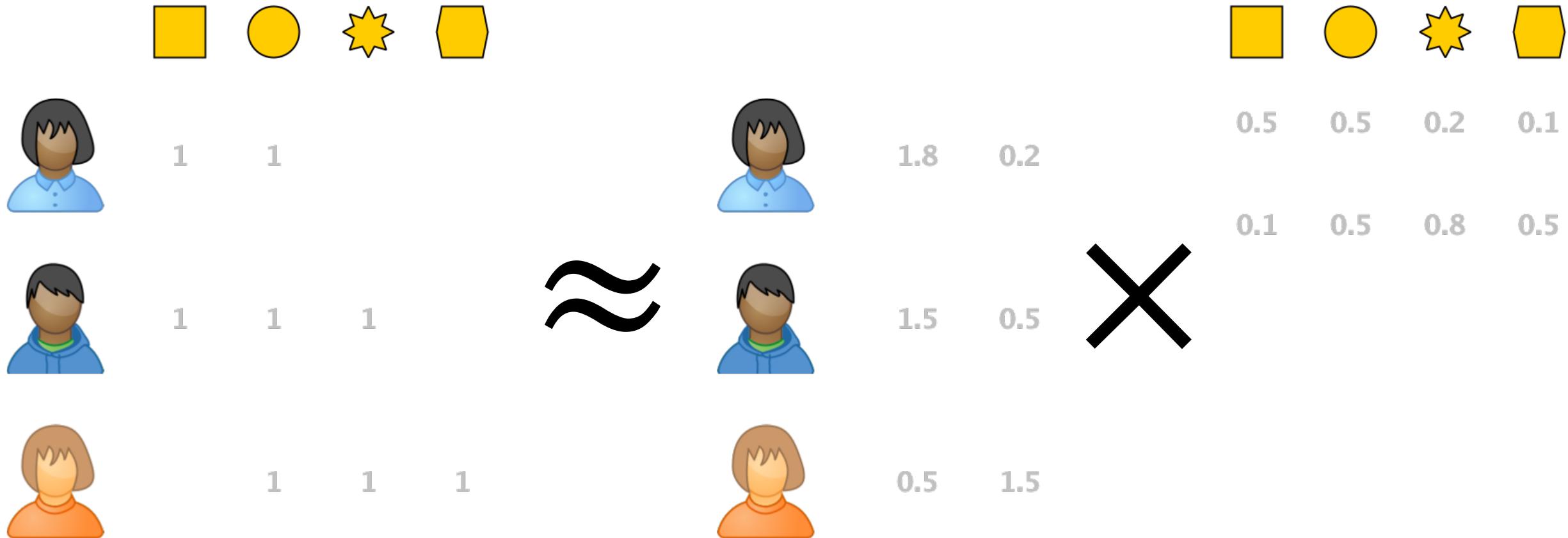
RecSys - ключевой  
драйвер бизнеса



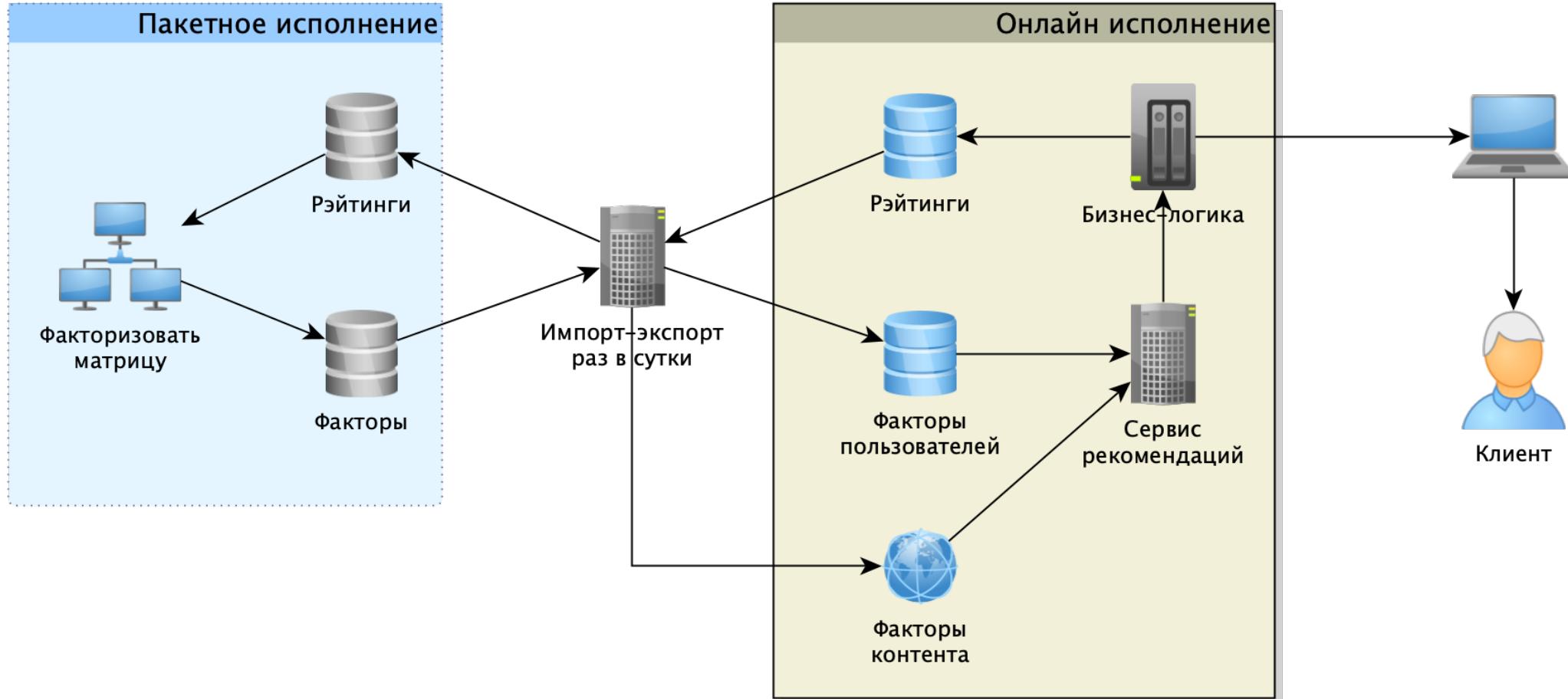
В приоритете цели  
бизнеса, а не цели  
пользователя



# Матрица знает, что вам подойдет



# 2010-е: Рекомендации в ПРОМ

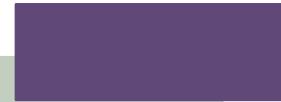


# Факторизация матриц в ПРОМ



## Плюсы

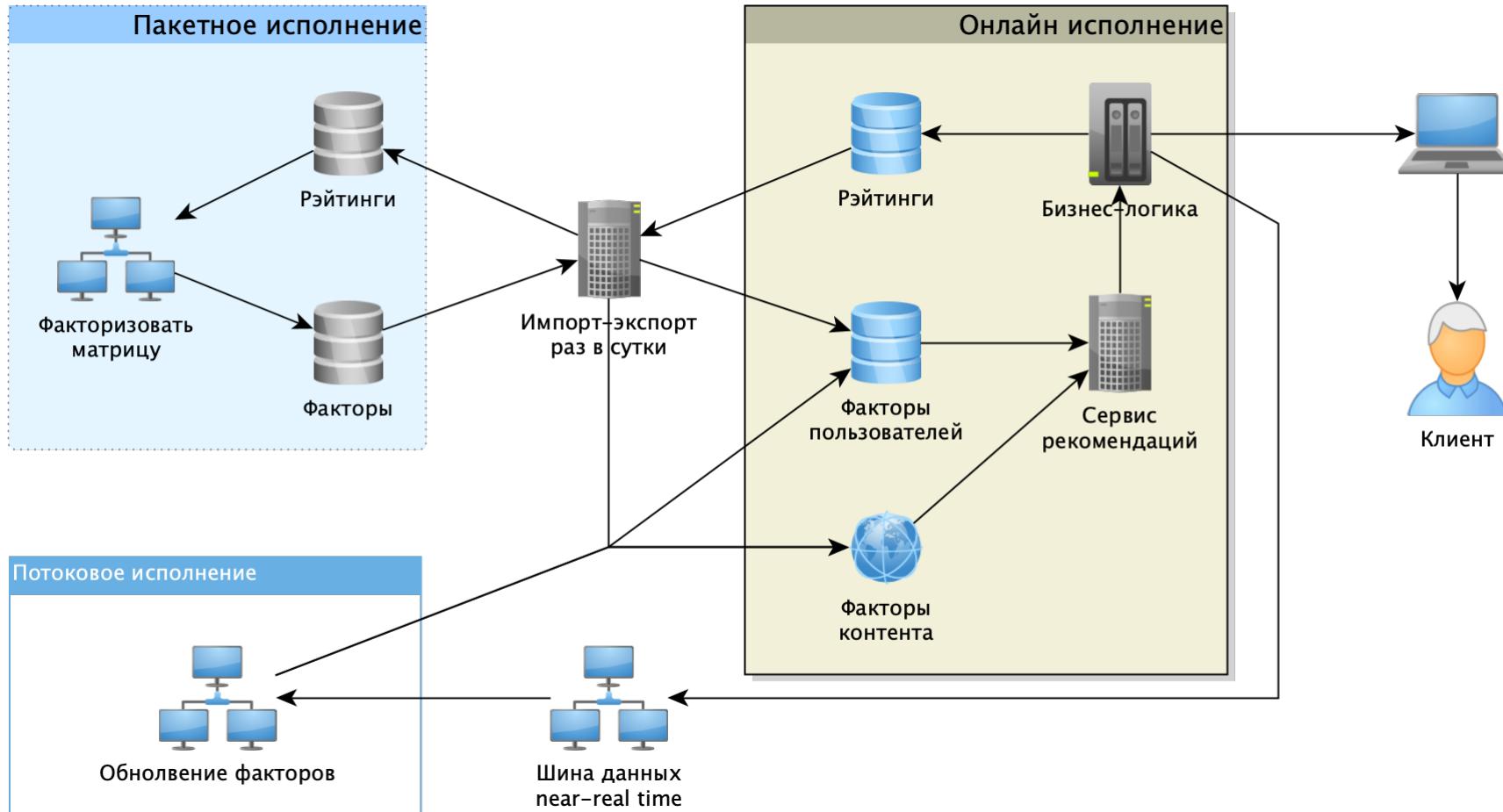
- Более выразительная модель
- Естественно раскладывается на части



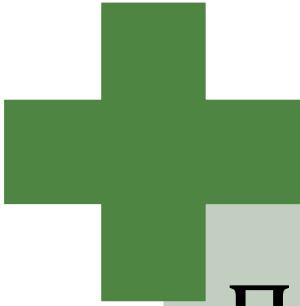
## Минусы

- Пропала онлайн реакция на обратную связь

# 2013-е: Рекомендации в ПРОМ



# Факторизация матриц в ПРОМ



## Плюсы

- Реакция на обратную связь в течении десятков секунд



## Минусы

- Несколько версий кода для потоковой и пакетной части

# Потоковый анализ изменил ланшафт промышленного МЛ

- Нет риска сложить ПРОМ (если неходить в ПРОМ базы)
- Достаточная свобода при выборе алгоритмов
- Реакция от нескольких секунд до нескольких минут
- Но нужны механизмы доставки данных из потокового обработчика в онлайн