ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

Лекция 5

Объединение таблиц: merge и concat

Оспан Ә.Ғ., PhD, доцент-исследователь

КазНУ, кафедра ИИ и Big Data

2026-2027 учебный год

Цель и содержание лекции

Цель и результаты обучения

- Выбирать ключи, тип merge и ожидаемую кардинальность.
- Находить несопоставленные ключи и размноженные события.
- Собирать части через concat с проверкой схемы и индекса.

Содержание лекции

1. **Ключи**
2. **Виды merge**
3. **Кардинальность**
4. **Конфликты**
5. **concat и файлы**

Журнал и справочник хранят разную информацию

Журнал событий

- Повторяющийся ключ объекта допустим.
- Событие имеет собственный идентификатор.
- Одна модель может участвовать во многих событиях.

Справочник объектов

- Ключ объекта обычно уникален.
- Хранит расшифровки и постоянные атрибуты.
- Несколько версий требуют явного правила выбора.

Уровень строки после присоединения справочника проверяется до любого суммирования.

Ключ и кардинальность задают контракт связи

Связь	Ключ слева	Ключ справа	validate
Один к одному	Уникален	Уникален	one_to_one
Многие к одному	Может повторяться	Уникален	many_to_one
Один ко многим	Уникален	Может повторяться	one_to_many
Многие ко многим	Повторяется	Повторяется	many_to_many

До merge называем смысл одной строки и ожидаемые кратности ключа.

left сохраняет события без соответствия

Пример Python

```
import pandas as pd

events = pd.DataFrame({
    'event_id': [101, 102, 103],
    'item_id': [1, 1, 3], 'amount': [10, 20, 30]
})
catalog = pd.DataFrame({
    'item_id': [1, 2, 4], 'label': ['Alpha', 'Beta', 'Delta']
})
joined = events.merge(
    catalog, on='item_id', how='left',
    validate='many_to_one', indicator=True
)
print(joined)
```

Результат: event_id 103: label отсутствует, _merge = left_only.

Сохранность

- Все три события остаются в результате.
- Ключ 3 получает пропуск описания.

Контракт

- on явно задает ключ.
- validate проверяет ожидаемую уникальность справа.

Четыре вида merge на одном примере

how	Что сохраняется	Строк в примере
left	Все левые события и их соответствия	3
right	Все правые объекты и их соответствия	4
inner	Только совпавшие ключи	2
outer	Ключи обеих сторон	5

Вид соединения выбирается по требуемому охвату, а не по удобному размеру.

indicator превращает соединение в аудит

1

Соединить

Явно задать on, how, validate и indicator=True.

2

Разделить статусы

both, left_only, right_only отвечают разным ситуациям.

3

Объяснить исключения

Неполный справочник, отсутствие событий или другая причина.

4

Сохранить аудит

Ключ, сторона и объяснение остаются проверяемыми.

- Правый объект без события не создает новую продажу.
- Пропущенный идентификатор не заменяем вымышленным нулевым ID.

Повторения ключа создают произведение кратностей

rows_for_key = n_left * n_right

Кратности

- n_left: число левых строк одного ключа.
- n_right: число правых соответствий.

Риск

- Каждая левая строка повторяется для правых версий.
- Суммы событий могут искусственно увеличиться.

2 левые строки * 3 правые строки = 6 комбинаций.

Результат many-to-many допустим только при содержательно обоснованной связи.

validate обнаруживает конфликт справочника

Пример Python

```
left = pd.DataFrame({'key': [1, 1], 'x': ['a', 'b']})
right = pd.DataFrame({'key': [1, 1, 1], 'y': [10, 20, 30]})
expanded = left.merge(right, on='key', how='inner')
print(len(expanded))
try:
    left.merge(right, on='key', validate='many_to_one')
except pd.errors.MergeError:
    print('Правый ключ не уникален')
```

Результат: 6; Правый ключ не уникален

Наблюдение

- Без контракта получается шесть строк.
- Число строк само не объясняет правильность связи.

Проверка

- many_to_one требует уникального правого ключа.
- Ошибка требует диагноза, а не подавления.

Повтор ключа: сначала диагноз, затем решение

1

Найти повторы

duplicated по явному ключу, keep=False.

2

Сравнить поля

Точные копии, разные версии или противоречия.

3

Определить правило

Подтвержденный дубль, актуальность на дату или отдельные сущности.

4

Повторить проверку

Уникальность, статусы соответствия и контрольные суммы.

- Не удаляем конфликтующие версии только ради успешного merge.
- Правило выбора должно быть обосновано до итоговой аналитики.

Похожий ключ еще не означает одинаковую сущность

Проверяем ключ

- Одинаковый предметный смысл с обеих сторон.
- Совместимые типы и важность ведущих нулей.
- Пустые ключи выделяются до соединения.

Проверяем атрибуты

- on задается явно, а не угадывается по общим именам.
- suffixes обозначают происхождение совпавших полей.
- Name не считается надежным ID игрока.

concat собирает строки частей журнала

Пример Python

```
part_a = events.iloc[:2].copy()
part_b = events.iloc[2:].copy()
all_events = pd.concat([part_a, part_b], ignore_index=True)
assert len(all_events) == len(part_a) + len(part_b)
assert all_events['event_id'].is_unique
print(all_events['amount'].sum())
```

Результат: 60

Назначение

- Сложить строки одинаковых по схеме частей.
- ignore_index создает новый технический индекс.

Контроль

- Число строк равно сумме частей.
- Ключи событий не должны неожиданно повторяться.

concat по столбцам выравнивает индекс

Операция	Основание соответствия	Основная проверка
merge	Ключевой столбец или набор полей	Смысл ключа и кардинальность
concat axis=0	Одинаковая схема столбцов	Пересечение частей и типы
concat axis=1	Индексные метки строк	Общий смысл и согласованность индекса
Простое равенство длины	Только число строк	Недостаточно для связи объектов

Несколько файлов: явный список и единая схема

Пример Python

```
from pathlib import Path

football = pd.read_csv(Path('data') / 'football.csv')
folder = Path('work_results') / 'lecture05_parts'
folder.mkdir(parents=True, exist_ok=True)
small = football[['Name', 'Age', 'Position']].head(12)
small.iloc[:6].to_csv(folder / 'part_01.csv', index=False)
small.iloc[6:].to_csv(folder / 'part_02.csv', index=False)
paths = sorted(folder.glob('part_*.csv'))
frames = [pd.read_csv(path) for path in paths]
assert frames
assert all(f.columns.tolist() == list(small) for f in frames)
combined = pd.concat(frames, ignore_index=True)
pd.testing.assert_frame_equal(combined, small)
```

Результат: Восстановлено 12 строк и 3 столбца.

Файлы

- Части создаются локально и имеют общую схему.
- Порядок файлов фиксирован сортировкой.

Сборка

- Сначала список таблиц, затем один concat.
- Сравнение подтверждает точное восстановление.

Проверки после объединения

merge

- Явные ключ, how и validate.
- Статусы соответствия и уникальность событий.

Суммы

- Контрольный объем до и после.
- Нет искусственного повторения вклада событий.

concat

- Схема, порядок файлов и пересечения частей.
- Индекс соответствует реальному правилу выравнивания.

Проверяйте сохранность событий по ключам и суммам, а не только размеру итоговой таблицы.

Пять правил объединения данных

1. Ключ объединения должен иметь одинаковый смысл в обеих таблицах.
2. left сохраняет левые строки без соответствия; inner их исключает.
3. Повторения ключа дают произведение кратностей, validate проверяет контракт.
4. Конфликт справочника исследуют до произвольного удаления версий.
5. concat требует согласованной схемы, индекса и контроля пересечений.

Проверьте ключи и сохранность событий

1. Что происходит с левой строкой без соответствия при left?
2. Сколько строк дают две левые и три правые записи одного ключа?
3. Как validate="many_to_one" защищает журнал событий?
4. Почему drop_duplicates по ключу не исправляет любой конфликт?
5. По чему выравнивает строки concat(axis=1)?

Самостоятельная работа

Создайте inner, который теряет одну строку и размножает другую, сохраняя общий размер. Докажите проблему через ID событий, indicator и контрольную сумму.

Источники по merge, concat и путям

Документация

1. pandas: merge и concat

https://pandas.pydata.org/docs/user_guide/merging.html

2. pandas: параметры DataFrame.merge

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.merge.html

3. Python: pathlib

https://docs.python.org/3/library/pathlib.html

Основа лекции

- Лекция 5 Объединение таблиц.pptx
- Лабораторная работа 5 .docx
- Лабораторная работа 5 с ответами.docx