일반 납품 - 세부 로직 (트럭 기준)

문서 설명

- 트럭 납품을 대한 설계 문서임
  - 기차, 비행기 납품은 전체적인 틀은 같지만, Step이나 세부 수치 등이 일부 다를 수 있음

의도

- 미래 생산량이 아닌 현재 보유량에 기반함
  - 유저의 미래 생산량을 예측하는 것은 불가능에 가까움 (유저별 실력, 세션 단절 등 때문에)
  - 타운십도 이런 이유에서 자원 보유량 기반 밸런싱을 했다고 알려짐
- 자원 보유량은 난이도 설정에는 영향을 끼치지 않음
  - 납품 대상으로 선정될 확률(가중치), 요구 수량에 영향을 끼침

목차

Step 1: 생산품 계층 태깅
Step 2: 획득처별 보유량 파악
Step 3: 자원 확보 상태 태깅
Step 4: 자원 동원 Case 결정
Step 5: 생산 압박 분산
Step 6: 중복 납품 방지
    Step 6.1: 직접 제한
    Step 6.2: 재료 제한 (To-do list 1순위)
Step 7: 납품 패턴 결정
Step 8: 납품 대상 선정 (보유량 기반)
Step 9: 납품 대상별 요구 개수 결정
Step 10: 선정 대상 제외 및 이전 Step 반복

Step 1: 생산품 계층 태깅

- 태그 분류: Top, Mid, Crops
- 분류 방법
  - Crops: 밭에서 생산하는 모든 작물
  - Top: 현재 해금된 레시피 기준으로 재료로 안 쓰이는 생산품 (작물 제외)
  - Mid: Crops, Top에 해당되지 않는 모든 생산품 (중간 재료 의미)

Step 2: 획득처별 보유량 파악

- 스냅샷으로 가상 창고 생성 시 각 아이템의 획득처별 보유량을 구분
  - 구조예시: Milk: {storage_qty: 5, shelf_qty: 2, market_qty: 3}

Step 3: 자원 확보 상태 태깅

- 태깅 방법: 획득처별 보유량을 기반으로 아래 우선순위에 따라 단 하나의 자원 확보 상태 태그 부여

- 1순위 태그: SOURCE_STORAGE
  - 조건: storage_qty가 1 이상

- 2순위 태그: SOURCE_SHELF
  - 조건: 1순위 만족 못하면서 shelf_qty가 1 이상

- 2순위 태그: SOURCE_MARKET
  - 조건: 1, 2순위 조건 만족하지 못하면서 market_qty가 1 이상

- 3순위 태그: SOURCE_PRODUCTION
  - 조건: 1, 2, 3순위 조건 모두 만족 못함 (보유량 0 의미)

- 자완 확보 태깅 데이터 예시
  - SOURCE_STORAGE: {Milk, Cream}
  - SOURCE_SHELF: {Soup}
  - SOURCE_MARKET: {Bread, Wheat}
  - SOURCE_PRODUCTION: {Carrot, Cookie}

Step 4: 자원 동원 Case 결정

- 납품에 필요한 자원을 어디서 획득하여 처리하는 가에 따라 4가지 Case가 나눠짐

- Case 1: 창고만으로 처리 가능
  - SOURCE_STORAGE로 태깅된 자원만 사용

    - storage_qty보다 적은 수량만 요구 가능
- Case 2: 창고 + 선반 및 수확 가능한 작물로 처리 가능
    - SOURCE_STORAGE, SOURCE_SHELF로 태깅된 자원만 사용
    - storage_qty + shelf_qty보다 적은 수량만 요구 가능
- Case 3: 창고 + 선반 및 수확 가능한 작물 + 시장으로 처리 가능
    - SOURCE_STORAGE, SOURCE_SHELF, SOURCE_MARKET로 태깅된 자원 사용 가능
    - storage_qty + shelf_qty + market_qty보다 적은 수량만 요구 가능
- Case 4: 즉각 처리 불가
    - 모든 자원 확보 태깅 사용 가능
    - storage_qty + shelf_qty + market_qty + production_qty보다 적은 수량만 요구 가능
        - production_qty는 생산 압박 분산에서 계산

- 관련 시트: DeliverySourcePolicy
    - 현재 어려움 지수에 비례하여 Case 별 가중치 결정
    - DeliveryType: 적용할 납품 종류
    - MinStruggleScore, MaxStruggleScore: 어려움 지수 범위
    - Storage, Shelf, Market, Production: 각 case의 가중치

- 예외 처리
    - 생산 압박 분산, 중복 납품 방지 등으로 납품 생성 실패 시 Case 순서대로 납품 재생성 시도
        - Case 1에서 첫 시도한 예시: Case 1 → 2 → 3 → 4 순으로 재시도
        - Case 2에서 첫 시도한 예시: Case 2 → 3 → 4 순으로 재시도

Step 5: 생산 압박 분산

- 기본 개념
    - 의도: 특정 생산 건물에 생산 부하가 몰리는 것 방지
    - 생산 압박 정의: 납품 요구로 인해 각 생산 건물에 발생하는 총 슬롯 부담
    - 계산식: 공장 당 임계치 * 동일 공장 수 + 창고 보유량 - 납품 요구량
    - 예시
        - 조건
            - 외양간: 2개 (총 12슬롯)
            - 창고에 우유 1개 보유

        - 납품1: 우유 3개 필요
        - 납품2: 우유 2개 필요
    - 계산
        - 외양간의 임계치: 6 (Bulding 시트의 PressureThreshold 컬럼 값)
        - 외양간 수: 2개
        - 우유 창고 보유량: 1개
        - 총 우유 수요: 5개
        - 남은 슬롯 수: 6 * 2 + 1 - 5 = 8 슬롯

<details>
<summary>상세 개념 및 예외 처리 (필독)</summary>

- 계층 부하 적용
    - 최종 생산품 뿐만 아니라 하위 재료 생산까지 고려해서 관련된 모든 생산 공장에 슬롯 부하 전파
    - 예시: 당근 수프 1개 (모든 하위 재료 없다고 가정)
        - 수프 공장: 1 슬롯
        - 유제품 공장(크림 생산): 1 슬롯
        - 외양간(우유 생산): 1 슬롯
        - 사료 공장(소 사료): 1/3 슬롯 (한번에 3개 생산)
        - 밭(밀/옥수수/당근): 7 + 2/3 슬롯 (수프에 밀 5, 당근 2, 사료에 밀 1/3, 옥수수1/3)
- 이미 생산 대기열에 다른 물품이 있을 때 대처 방안
    - 가정
        - 공장 1에서 물품 A, B, C 생산 가능
        - 납품에서 A, B만 요구
        - 공장 1의 생산 대기열에는 C가 6개 존재
        - 여유 슬롯이 3개 있다고 판정 (대상 아닌 거 1개당 0.5 슬롯 취급)
            - 단, 0.5라는 계수는 고정 값이 아닌 config임 (QUEUE_LOAD_FACTOR)
        - 당연하게도 납품에서 C를 1개 요구하면 압박치는 3에서 3.5로 상승함
            - C 1개 요구 전: 0.5 * 6 = 3
            - C 1개 요구 후: 0.5 * 5 + 1 = 3.5
        - 만약 납품에서 C를 3개 요구하고 대기열이 비어있는 상태에서 C를 6개 넣으면?
            - 3 + 0.5 * 3 = 4.5로 계산
- 생산처별 압박 임계치
    - 기본적으로 Building 시트의 PressureThreshold 컬럼 참조
</details>

- 밭은 해당 레벨에서 해금된 밭의 수로 설정
  - BuildingInstance 시트에서 UnlockLevel과 MaxAmount 참조

<details>
<summary>종합 예시</summary>

- 예시1: 외양간은 우유만 생산하고 최대 임계치가 12, 현재 임계치가 4라면 남은 여유는 8
  -> Milk: 8
- 예시2: 유제품 공장의 최대 임계치 6, 현재 임계치가 0이라면 크림의 여유는 6
  -> Cream: 6 (크림의 재료는 우유 1개)
- 예시3: 유제품 공장의 최대 임계치 6, 현재 임계치가 0이라면 치즈의 여유는 4
  - 우유 여유분이 8이라서 유제품 공장이 여유가 있어도 6까지 상승 불가
    -> Cheese: 4 (치즈의 재료는 우유 2개)
- 예시4
  - 빵 공장의 최대 임계치 6, 빵 2개와 쿠키 1개 요구 중이면 현재 임계치는 3
  - 밀과 달걀은 충분하다고 가정
    -> Bread: 3
    → Cookie: 3

</details>

- 획득처별 보유량에 신규 속성 추가 (데이터 예시)
  - 생산 압박 계산 전: Milk: {storage_qty: 5, shelf_qty: 2, market_qty: 3}
  - 생산 압박 계산 후: Milk: {storage_qty: 5, shelf_qty: 2, market_qty: 3, production_qty: 8}

- **To-do list: 생산 압박 완전 회피**
  - 납품 내 다양성 확보가 부족한 경우 개발
  - 동일 공장 소속 레시피 중 하나만 각 납품에 선정 가능
    - 예시: 빵 공장에서 빵과 쿠키 생산 가능, 납품 하나의 관점에서 빵과 쿠키 중 한 종류만 참여 가능

Step 6: 중복 납품 방지

- 기본 개념 및 의도
  - 최근 납품한 물품을 또 납품하지 않도록 조정
  - 최근 N회 납품 중에서 특정 생산품이 m회 이상 등장했다면 해당 생산품을 납품 대상으로 선정하지 못하게 함

- 2가지 세부 로직
  - 직접 제한: 특정 생산품이 납품에 출현하는 빈도를 직접적으로 제어
  - 재료 제한: 특정 재료가 많이 사용되면 해당 재료를 사용하는 물품의 납품 빈도 제어 (생산 루프 다양화)

Step 6.1: 직접 제한

- 3가지 고려 요소
  - 생성 히스토리: 최근 생성된 N회 납품에서 동일 생산품은 m회 까지만 선정 가능
  - 완료 히스토리: 최근 완료한 N회 납품에서 특정 생산품이 m회 이상 등장했다면 새 납품에 선정 금지
  - 현재 생성된 납품 고려: 현재 생성된 납품에 특정 생산품이 m회 이상 등장한 상태면 새 납품에 선정 금지

- 관련 시트: DeliveryRepeatLimit
  - DeliveryType: 적용할 납품 종류
  - MinLevel, MaxLevel 컬럼: 적용 레벨 범위
  - RepeatScopeType: 3가지 고려 요소(생성 히스토리, 완료 히스토리, 현재 생성된 납품) 중 무엇인지 의미
  - HistoryWindow: 히스토리를 몇 개 까지 참조할지 정의, RepeatScopeType이 Active면 빈 값
  - ThresholdList: 히스토리 내 등장한 횟수
  - PenaltyFactorList: 등장 횟수에 따른 페널티 계수
  - 예시
    - RepeatScopeType: Created
    - HistoryWindow: 9
    - ThresholdList: 1 / 3
    - PenaltyFactorList : 0.2 / 0
    - 최근 생성된 9개의 납품에 동일한 생산품이 1번 출현하면 페널티 계수 0.2, 3번 이상 출현하면 0
      (정확히는 1번 이상 3번 미만, 3번 이상으로 처리해야 함)

- 페널티 계수 적용
  - 3가지 고려 요소 중 가장 낮은 페널티 계수를 Step 7에서 적용

Step 6.2: 재료 제한 (To-do list 1순위)

- 생산 압박 분산, 직접 제한으로 생산 루프 다양성 확보가 불가능하면 개발

<details>
<summary>상세 내용 (개발 시 필독)</summary>

- 3가지 고려 요소
  - 직접 제한과 동일하게 생성 히스토리, 완료 히스토리, 현재 생성된 납품 고려
- 관련 시트: DeliveryIngredientPenalty
  - DeliveryType, MinLevel, MaxLevel, RepeatScopeType, HistoryWindow: 동일 의미
  - Item: 적용 대상 Item (직접 제한과 달리 레벨 뿐만 아니라 아이템 까지 구분해서 적용함)
  - ThresholdList, PenaltyFactorList: 동일한 역할
- 페널티 계수 적용
  - 3가지 고료 요소 중 가장 낮은 페널티 계수를 Step 7에서 적용
  - (Step 7 자체 값 * 직접 제한 페널티 계수 * 재료 제한 페널티 계수

</details>

Step 7: 납품 패턴 결정

- 납품 패턴 설명
  - 각 납품 패턴은 요구할 생산품 종류 수, 생산품별 요구량 범위가 정의되있음 (간혹 생산품 계층까지 지정)
  - 납품 패턴별로 출현 가중치와 난이도가 정해져 있음

- 선택 가능한 납품 패턴 판별 1: 종류 기반
  - 이전 Step들을 통해 납품에 선정될 수 있는 생산품 리스트 확보가 가능해짐
    - 자원 동원 Case, 생산 압박 분산, 중복 납품 방지 과정을 거쳐 납품 가능 생산품 리스트를 만들 수 있음
  - 만약, 리스트 내에 생산품이 5종 뿐이라면 6종을 요구하는 납품 패턴은 가중치 계산에서 제외
    - 리스트 내 Top 계층이 2종 뿐이면 Top 계층 3종 이상 요구하는 납품 패턴도 가중치 계산에서 제외
- 선택 가능한 납품 패턴 판별 2: 개수 기반
  - 최소 요구치를 기반으로 가능 여부 판단
  - **주의 사항: 태깅만 맞는 보유량만 고려해야 함**
  - 예시1: 3종 요구 / 요구 스코어 최소치: 100, 200, 300

        - 보유량 * 스코어가 300이 넘는 물품이 있는지 확인
          → 없으면 불가능 패턴
          → 있으면 해당 물품 제외하고 다음 스코어 확인
        - 보유량 * 스코어가 200이 넘는 물품이 있는지 확인
          → 동일 로직으로 판별 → 마지막 스코어 까지 쭉 확인
    - 예시2: 3종 요구 요구 스코어 최소치: 200, 200, 200
        - 한번에 200 넘는 거 3개 이상인지 확인하거나 위 예시처럼 그냥 순차적으로 처리
- 예외 처리
    - 선정 가능한 패턴이 없으면 다음 순서인 자원 동원 Case 기준으로 다시 납품 패턴 선정
        - Case 1 → 2 → 3 → 4 순으로 재시도
    - 만약 Case 4도 실패하면 생산 압박 분산, 중복 납품 방지를 무시하고 Case 4로 재시도

- 관련 시트: DeliveryPattern
    - DeliveryType: 적용할 납품 종류
    - **MinLevel, MaxLevel 컬럼: 출현 레벨 범위**
        - **다른 시트와 다르게 범위 중복됨**
    - LayerList: 요구할 생산품의 계층, 빈 값이면 계층 제한 없음
    - FixedAmountList: 동일 인덱스의 LayerList 물품을 몇 개 요구 할지 정의
    - MinScoreList, MaxScoreList: 동일 인덱스의 LayerList 물품을 최초/최대 요구량을 Score 기반으로 정의
        - 정확한 개수는 Step 9 참조
        - FixedAmount, Min/MaxAmount, Min/MaxScore 중에 한 쪽만 사용함
    - Weight: 해당 납품 스코어의 가중치 (동일 그룹 내에서 가중치 계산)
    - SplitCount: 기차 납품 전용 (기차 납품 문서에서 설명: [11차 스프린트] 기차 납품)

- 납품 패턴 고도화 (**To-do list 2순위**)
    - 어려움 지수에 따른 납품 패턴 가중치 변경
    - 시트 변경: DeliveryPattern 시트에 난이도 컬럼을 추가 (Very Easy ~ Very Hard)
    - 시트 추가: DifficultyFactorPolicy
        - DeliveryType: 적용할 납품 종류
        - MinStruggleScore, MaxStruggleScore: 어려움 지수 범위
        - VeryEasy~VeryHard: 난이도별 가중치 계수

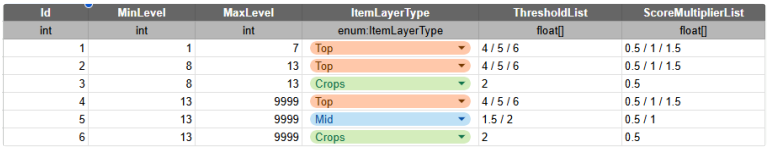
Step 8: 납품 대상 선정 (보유량 기반)

- 자원 보유량과 중복 납품 방지의 페널티 계수 등에 기반하여 납품 대상 선정

- **주의 사항**: Min/MaxScore 패턴인 경우 가장 높은 score 해당하는 물품부터 선정
  - 해당 요구 score를 충족하는 물품만 납품 대상 가능 리스트로 취급

- 용어 설명: 자원 보유량
  - 자원 보유량은 storage_qty + shelf_qty임
- 용어 설명: 잠재 수요 총량
  - 현재 해금된 레시피 기준으로 모든 아이템을 1개씩 만들 때 각 아이템이 필요한 개수 (자기 자신도 포함)
  - 예시: 우유
    - 크림에 우유 1개, 치즈에 우유 2개 소모
    - 당근 수프가 크림을 1개 소모
    - 우유 자체도 1개
    - 합계: 1+2+1+1=5

- 납품 대상 선정 기본 가중치
  - 기본 가중치는 잠재 수요 총량의 역수임
- 페널티 계수 적용
  - 중복 납품 방지의 직접 제한에서 계산한 페널티 계수를 기본 가중치에 곱함
    -> 기존 가중치 * 직접 제한 페널티 계수
- 자원 보유량 임계치 정의
  - 자원 보유량이 일정량 이상이면 가중치에 자원 보유 계수 적용
  - 임계치는 잠재 수요 총량의 배수임
  - 예시
    - 고계층 30종, 중간 재료 10종, 작물 10종 존재
    - 중간 재료 1종이 임계치에 도달하면 가중치 증가 (동일 계층 종류 수 비례)
    - 고계층 2종이 임계치에 도달하면 가중치 증가 (동일 계층 종류 수 비례)

- 관련 시트: DeliveryTargetThreshold
  - DeliveryType: 적용할 납품 종류
  - MinLevel, MaxLevel: 적용 레벨 범위
  - ItemLayerType: 고계층, 중간 재료, 작물 구분용
  - ThresholdList: 각 인덱스는 잠재 수요 총량의 배수를 의미 (임계치)
  - ScoreMultiplierList: 각 임계치에 도달했을 때 계수 정의
  - 추가 설명 및 예시
    - 

Id	MinLevel	MaxLevel	ItemLayerType	ThresholdList	ScoreMultiplierList
int	int	int	enum:ItemLayerType	float[]	float[]
1	1	7	Top	4 / 5 / 6	0.5 / 1 / 1.5
2	8	13	Top	4 / 5 / 6	0.5 / 1 / 1.5
3	8	13	Crops	2	0.5
4	13	9999	Top	4 / 5 / 6	0.5 / 1 / 1.5
5	13	9999	Mid	1.5 / 2	0.5 / 1
6	13	9999	Crops	2	0.5

    - 고계층 30종이라고 가정, 첫 번째 row 기준으로 설명
    - 잠재 수요 4배에 도달하면 동일 계층 종류 수의 0.5배 만큼 가중치 곱합
      -> 30종이므로 가중치 15배 의미

- 중복 선정 방지
  - 납품 1개 구성에 여러 개 물품이 필요한 경우가 많음
  - 납품 대상 선정 시 이미 선정한 물품은 제외 필요

- To-do list
  - 계층별 계수: 필요 시 계층별 가중치 계수 추가
  - 자원 보유량 정의 고도화
    - market_qty도 고려
    - 자원 획득처 별로 계수 적용: storage_qty + shelf_qty * 계수 + market_qty * 계수
      - 선반 계수: DELIVERY_TARGET_SELECTION_FACTOR_SHELF
      - 시장 계수: DELIVERY_TARGET_SELECTION_FACTOR_MARKET

Step 9: 납품 대상별 요구 개수 결정

- 해당 Step은 아래 조건을 만족해야 개수 결정에 관여함
  - Step 7: 납품 패턴 결정에서 FixedAmountList가 아닌 Min/MaxScoreList 쪽을 선택한 경우

- 기본 동작 방법
  - MinScore 이상을 충족하도록 최소치 보장

- MaxScore 초과하지 않도록 최대치 제한
- Min/MaxSocretList에 따라 가능한 경우를 동일 가중치로 계산하여 선정
- 예시: Min이 2, Max가 4라면 2, 3, 4를 동일 가중치로 계산해서 선정

- 예외처리: 자원 동원 Case 우선
  - Min, Max 범위 중에서 불가능한 case는 제외하고 가중치 계산해야함
  - 이전 Step에서 결정한 자원 동원 Case와 획득처별 보유량을 기반으로 가능한 case만 가중치 계산에 포함
  - 예시1
    - Min: 2, Max: 4
    - 자원 동원 Case : SOURCE_STORAGE
    - 획득처별 보유량 예시: storage_qty: 3, shelf_qty: 0, market_qty: 0, production: 8
    - 이 경우 실제 선택 가능한 수치는 2, 3, 4가 아니라 2, 3임
  - 예시2
    - Min: 2, Max: 4
    - 자원 동원 Case : SOURCE_STORAGE
    - 획득처별 보유량 예시: storage_qty: 1, shelf_qty: 0, market_qty: 0, production: 8
    - 이 경우 실제 선택 가능한 수치는 2, 3, 4가 아니라 1임
    - 단, 어떤 경우에도 1 이하로는 줄어들지 않음

Step 10: 선정 대상 제외 및 이전 Step 반복

- 선정 대상 제외
  - Step 6(중복 납품 방지) 까지 수행하면 납품 가능 리스트가 확보됨
  - Step 8(납품 대상 선정)에서 선정한 대상을 납품 가능 리스트에서 제외
- 이전 Step 반복 및 납품 완료
  - 납품에서 요구하는 아이템 종류 수를 충족하지 못했다면 다시 Step 8~9를 반복
  - 요구하는 아이템 종류 수를 모두 채웠다면 납품 생성 완료

- 납품 생성 실패 시 예외 처리
  - 납품 생성 실패 원인: 선택된 자원 동원 Case로는 생산 압박 분산/중복 납품 방지를 통과해 납품 패턴 선정이 불가능한 경우

- 납품 생성이 불가능한 경우 : 자원 동원 Case가 SOURCE_PRODUCTION인데도 납품 생성에 실패한 경우
  - 이 경우 생산 압박 분산/중복 납품 방지를 무시하고 납품 패턴 선정