# 기획서

<https://www.notion.so/_-3bd81f8546484ba2a123ff3666d9fdc5#e011cc2f594f4ac9b495716086784c39>

* 업적 활성화 여부
  + **Q.** 활성화가 꺼져 있어도 누적하고 있을 것인가?
  + A. 클라이언트에서만 쓰는 표시 유무 이므로 서버는 누적하고 있어야 함.
* **업적 적용 범위**
  + 캐릭터 기준 : 한 서버 내 속한 캐릭터 1명
  + 서버 기준 : 한 서버 내 속한 캐릭터 모두
  + 계정 기준 (현재 개발 사항 없음) : 모든 서버의 캐릭터 모두
* 업적 획득 유저 수 표시 여부
* 업적 획득 등수 표시 여부
* 동시성 처리에 대한 의문???
* 업적 진행도 노트 간격이란??
* 위업

## 2/14 공유

@D.손봉호 님, 서버 내 예외 처리 사항들 정리하여 공유 드립니다. 상세 내용은 하단 노션 문서 참고 부탁 드립니다.

관련된 시스템들이 아직 개발되지 않았기에 추후 상황에 따라 변경될 가능성 매우 높은 점 염두 부탁 드립니다.

구현상 문제가 있는 부분이나 정책 수정이 필요한 부분은 편하신 경로로 문의 부탁 드립니다.

시작 시 들고 시작할 업적 관련된 내용은 논의 되는 대로 공유 드리겠습니다 (\_ \_)

<https://www.notion.so/_-3bd81f8546484ba2a123ff3666d9fdc5?pvs=4#befc97b7292b42c8bcfd7612229325d4>

[도전업적]

목표

불필요한 CS 인입 최소화

방향

업적 통합 과정에서 유저 별도 액션 없이, 더 높은 진행도의 업적으로 덮어쓰기 처리

(업적 진행 카운트 통합은 배제)

[위업]

목표

위업 달성자 변경 시 중복 재화 지급 및 어뷰징 방지

방향

달성자가 변경될 때마다 이전 지급자에게 지급된 보상을 회수 후 신규 달성자에게 지급하여 재화 중복 지급 방지

회수가 어렵거나 어뷰징이 예상될 경우 칭호 등 인게임 재화와 무관한 별도 아이템으로만 한정 지급

## 2/16 공유

1. 업적 (누적형 Objective)는 통계적 정의에 기반한 것들만 개발

예를 들어 담당자가 달라 비통계적 개념까지 Objective로 개발 요청이 오면 저에게 노티해주세요.

1. 위업은 일회성, 누적형, 퀘스트, 업적 등 모든 형태가 될 수 있음 (World 기준)

업적 메인 타입과 서브타입은 몇 백개 이상씩 개발될 수 있음

1. 메인만 몇 백개 가능 메인X서브는 몇 천개 가능
2. 업적이나 위업은 활성화되는 개방 조건(ActiveCondition)이 별도 개념으로 추가될 수 있음
3. 퀘스트 발급 조건, 업적 개방 조건 등등의 형태
4. 업적 통합관리 형태와 비슷하게 이 조건들도 통합 관리 되길 희망

차후에 구체적인 예시를 통해 개발 설명 드릴 예정

중요한 내용이니 이견 없으시면 이모지 반응 한번 부탁드립니다.

감사합니다. (편집됨)

수정 기획링크:

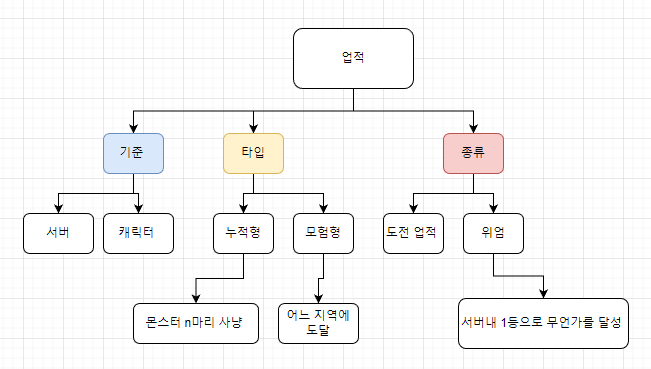
<https://www.notion.so/_-3bd81f8546484ba2a123ff3666d9fdc5#87608a69eae24ecc9ca2e6ae1f85acb8>

## 3/14 공유

1. 최초 달성자만 업적 달성자로 전체 알림
2. 직업 변경에 따른 레벨 업적은 어떻게 할지? <- 기획에서 논의 중
3. 서버 내 다른 유저들에게 지급된 보상은 회수되지 않아도 무방합니다

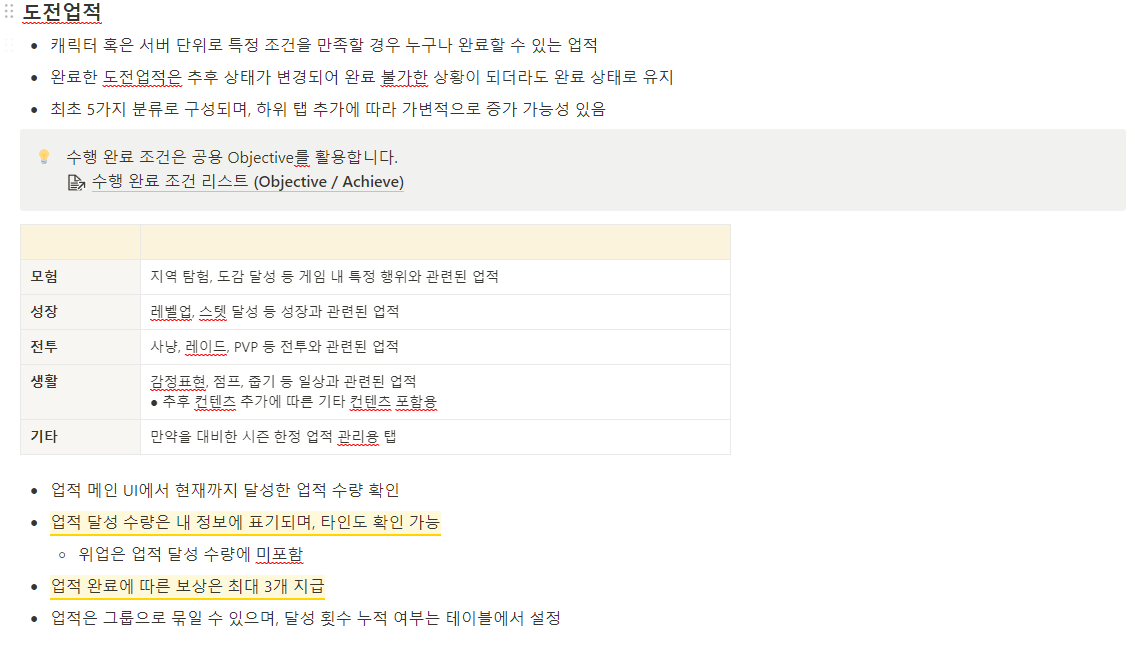
## ObjectiveType:

<https://www.notion.so/Objective-Achieve-c242ed5266e54c00854930f89875e587>

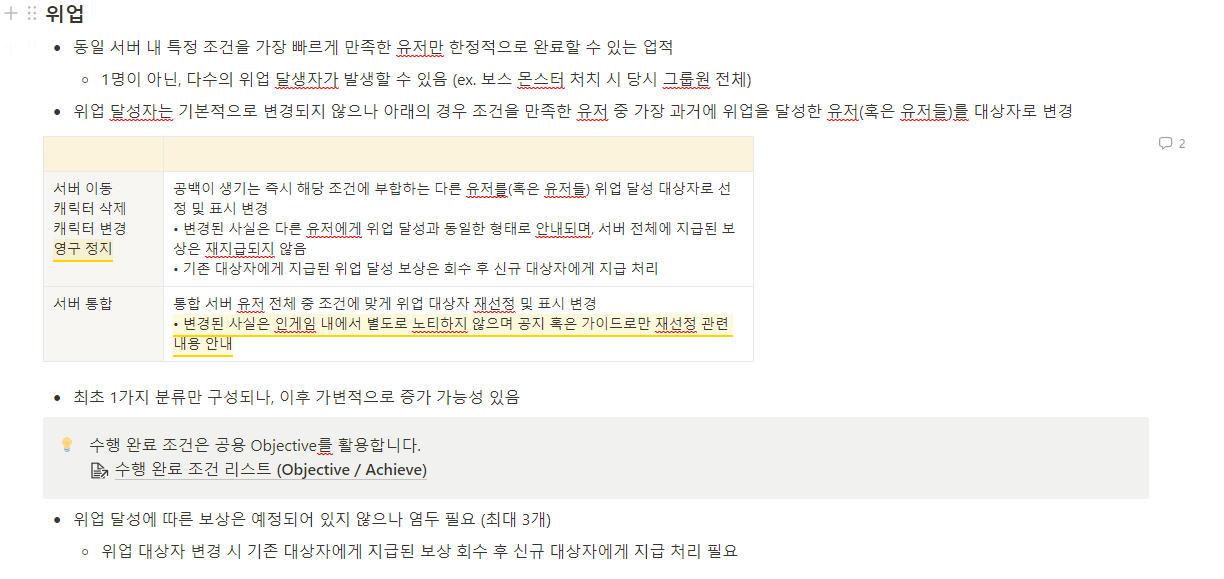


# 업적 종류

## 도전 업적



## 위엄



## 이벤트 업적(???)

* “바람의 나라:연” 아쉬운 점 모은 부분에서 발견

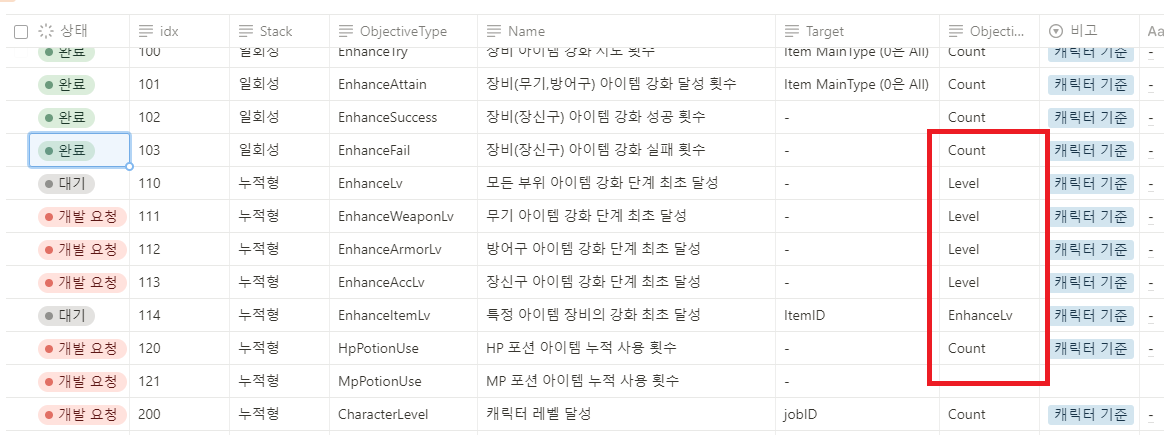
# 소유

* 캐릭터 ,계정, **서버(도전 업적)** 기준

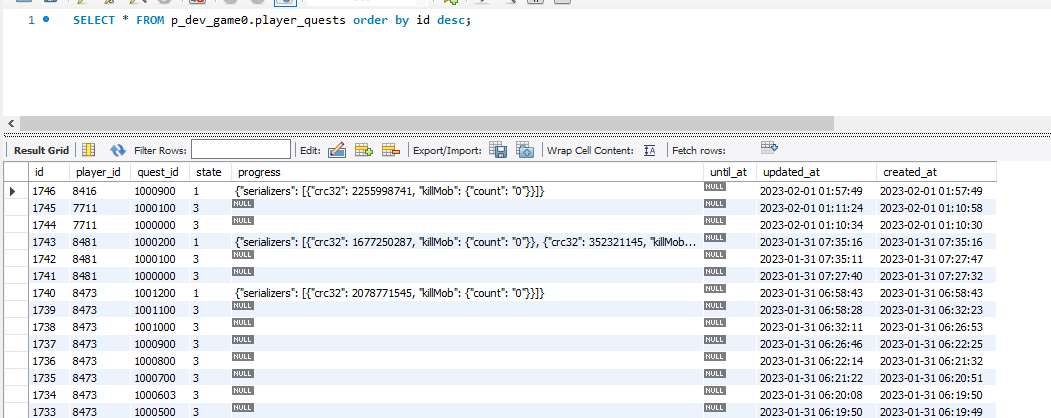


# 

# Objective Type



* ex) 퀘스트 db 관련

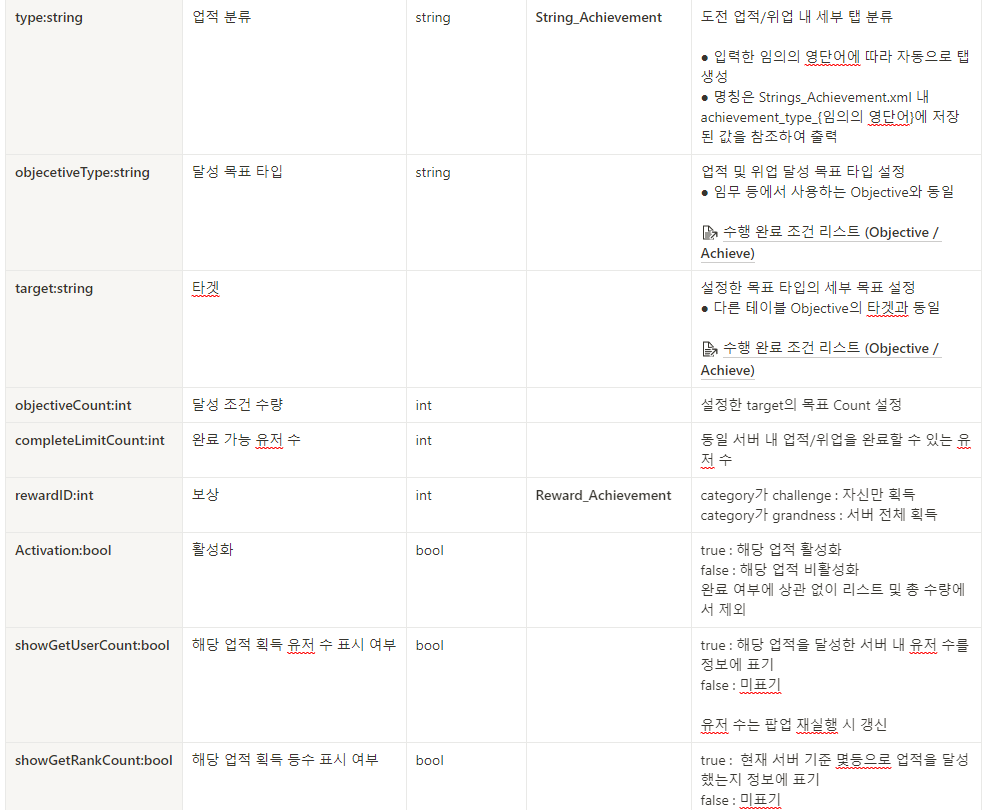


* **업적에 타입**
* (캐릭터, 계정) 누적 count
* (캐릭터, 계정) 목표 달성 방식
  + 특정 몬스터 사냥
  + 레벨 달성
  + 강화 레벨 도달
  + 특정 지역 탐험
  + 도감 달성
  + hp, mp, 공격력 n값 달성
  + 체력 n이하로 살아남기
* (캐릭터, 계정) 업적 순위??

# Excel 테이블

* 단위에 대해 있었으면?? (계정 인지? 캐릭터 인지?)





# 작업 방향 제안

## 퀘스트 시스템과 유사하게 제작

ex)퀘스트

| public static QuestProgress OfResQuest(ResourceSystem rSys, RQuest rQuest)  {  var rsz = new TRepeatedObjectiveSerializer();  var objectives = ImmutableObjectivesOf(rQuest, rsz);  return new QuestProgress(objectives);  } |
| --- |

=>

| public static AchivementProgress OfResAchivement(ResourceSystem rSys, RAchivement rAchivement)  {  var rsz = new TRepeatedObjectiveSerializer();  var objectives = ImmutableObjectivesOf(rAchivement, rsz);  return new AchivementProgress(objectives);  } |
| --- |

TRepeatedObjectiveSerializer : 일일/ 주간 업적 처럼 반복적인 퀘스트에 적용

TGoalObjectiveSerializer : 목표 달성형 업적에 적용

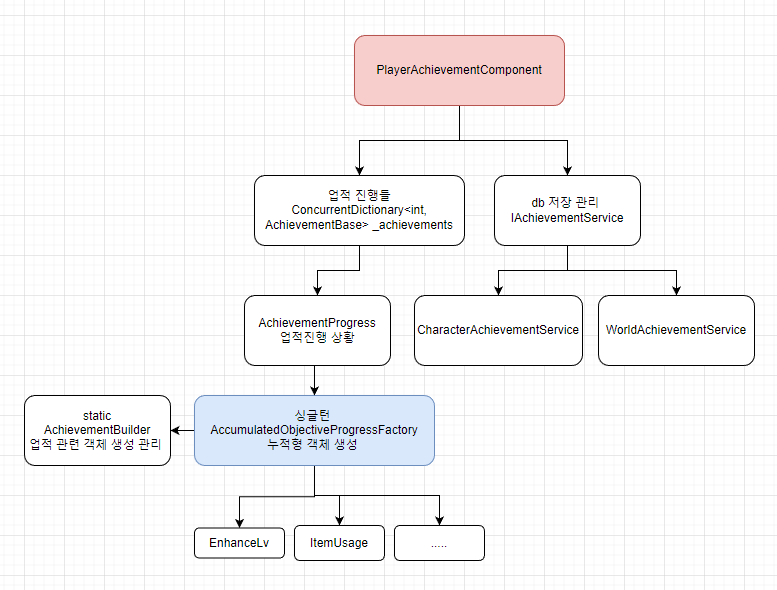
(ex.특정 몬스터 사냥/레벨 달성/강화 레벨 도달/특정 지역 탐험/도감 달성/hp, mp, 공격력 n값 달성/체력 n이하로 살아남기)

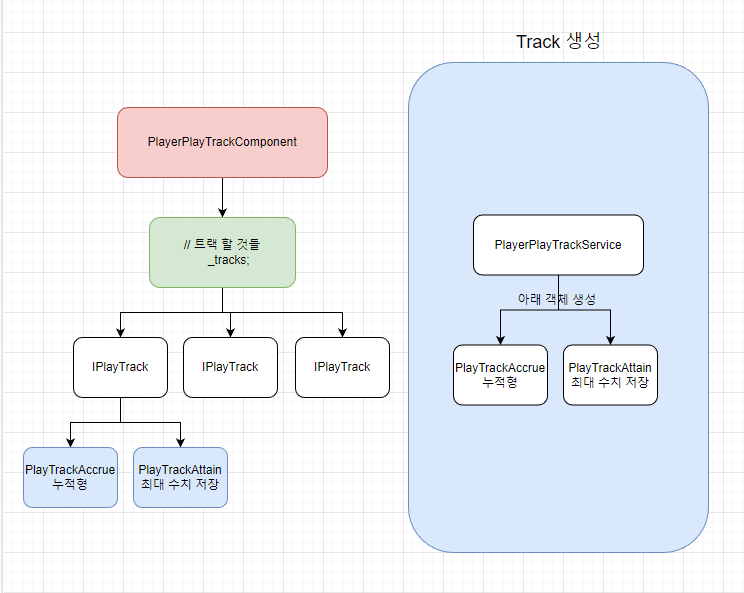
**참조) 바람의 나라:연 나무 위키 참조 -**

* <https://namu.wiki/w/%EB%B0%94%EB%9E%8C%EC%9D%98%EB%82%98%EB%9D%BC:%20%EC%97%B0>

| * **업적** 크게 일일 업적, 주간 업적, 도전 업적으로 나뉘며 일일 업적은 하루에 1번, 주간 업적은 일주일에 1번 초기화된다. 도전 업적은 모험, 도전, 생활로 나뉘는데 업적에 따라 난도가 천차만별이다. 완료하면 자잘한 보상과 칭호를 얻을 수 있다. 여담으로 이모티콘 업적을 빠르게 깨기 위해 한꺼번에 매크로에 등록하는 꼼수가 있었으나 2020년 7월 24일에 막혔다. |
| --- |

# 구성





# DB

| CREATE TABLE `player\_achievements` (  `id` bigint NOT NULL AUTO\_INCREMENT COMMENT '시퀸스',  `player\_id` bigint NOT NULL COMMENT '플레이어 아이디',   `achievement\_id` int NOT NULL COMMENT '업적 아이디',  `state` int NOT NULL DEFAULT '0' COMMENT '상태',  `progress` json DEFAULT NULL COMMENT '진행 데이터',  `until\_at` datetime DEFAULT NULL COMMENT '기간제일 경우',  `updated\_at` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '갱신일',  `created\_at` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '생성일',  PRIMARY KEY (`id`),  UNIQUE KEY `player\_achievements\_UN` (`player\_id`,`achievement\_id`) ) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci COMMENT='플레이어 업적'; |
| --- |

# GameAchievementSystem 추가

| public GameAchievementSystem Achievement { get; } |
| --- |

# 업적 관련 Factory

| private AccumulatedObjectiveProgressFactory()  {  var temp =  new Dictionary<AccumulatedObjectiveType, Func<RAccumulatedObjectiveProperty, TAchievementObjectiveSerializer, AccumulatedObjectiveProgressBase>>();  temp.Add(AccumulatedObjectiveType.EnhanceLv,  (res, progress) => new EnhanceLvObjectiveProgress(res, progress?.EnhanceLv));  temp.Add(AccumulatedObjectiveType.ItemUsage,  (res, progress) => new ItemUsageObjectiveProgress(res, progress?.ItemUsage));  temp.Add(AccumulatedObjectiveType.CharacterLevel,  (res, progress) => new PlayerLevelObjectiveProgress(res, progress?.PlayerLevel));  temp.Add(AccumulatedObjectiveType.CharacterDie,  (res, progress) => new PlayerDieObjectiveProgress(res, progress?.PlayerDie));  temp.Add(AccumulatedObjectiveType.AcquireGold,  (res, progress) => new CurrencyEarnedObjectiveProgress(res, progress?.Currency));  temp.Add(AccumulatedObjectiveType.MonsterKill,  (res, progress) => new MonsterKillObjectiveProgress(res, progress?.MonsterKill));  temp.Add(AccumulatedObjectiveType.QuestClear,  (res, progress) => new QuestStateObjectiveProgress(res, progress?.Quest));  temp.Add(AccumulatedObjectiveType.AttributeAttain,  (res, progress) => new AttrAttainObjectiveProgress(res, progress?.AttrAttain));  temp.Add(AccumulatedObjectiveType.TitleAttain,  (res, progress) => new TitleObjectiveProgress(res, progress?.Title));  temp.Add(AccumulatedObjectiveType.PoseAttempt,  (res, progress) => new PoseObjectiveProgress(res, progress?.Pose));    \_constructors = temp;  } |
| --- |

# 체크 필요 사항

* 업적 랭킹
* account\_uid 기반 db 처리
* GameAchievementSystem 에 대해
* 스텝이 올라가면 이전 스텝에 대한 정보로 우선 업데이트 기능**(완료)**

Q. CRC32는?

A. 이 값은 고유하여야 하므로 별도의 group을 만들어서 넣자.

# 최적화 방안

* 각 그룹 별 최대 index 를 저장하고 있고

| *MaxGroupStepIds = new List<int>();*  *ZeroGroupIds = new List<int>();*  *// 각 그룹별 최대 스텝 id*  MaxGroupStepIds = \_achievementsById.Values.Where(x => x.Spec.GroupID > 0)  .GroupBy(x => x.Spec.GroupID)  .ToDictionary(x => x.Key, x => x.Max(y => y.Spec.Id))  .Values.ToList();  *// groupidx=0 인 업적들*  ZeroGroupIds = \_achievementsById.Values.Where(x => x.Spec.GroupID <= 0)  .Select(x => x.Spec.Id).ToList(); |
| --- |

| *// 각 groupIndex에 최대 스텝을 가져오자*  var maxGroupStepIds = rSys.Achievement.MaxGroupStepIds;  *// DB에 업적에 최대 Step이 있으면 그 밑에 Step들은 모두 보상 받은 것이다.*  *// 이전에 받은 보상들*  var rewardAllGroups = new List<int>();  var playerGroups = db.Game.PlayerAchievement.GetByAchieveIdsAsync(playerUid, AchievementState.Awarded.Value, maxGroupStepIds).Result  .Select(k => rSys.Achievement.GetById(k.achievement\_id).Spec.GroupID);  rewardAllGroups.AddRange(playerGroups);  var worldGroups = db.Game.WorldPlayerAchievement.GetByAchieveIdsAsync(accountUid, playerUid, AchievementState.Awarded.Value, maxGroupStepIds).Result  .Select(k => rSys.Achievement.GetById(k.achievement\_id).Spec.GroupID);  rewardAllGroups.AddRange(worldGroups);  \_rewardAllGroups = rewardAllGroups.ToHashSet(); |
| --- |

# 리뷰

## 1차

1. 인챈트 비용 산정 하기 위한 스타틱 함수가 필요함**(완료)**

- 유니 테스트에서도 쓰기 위함

2. AchievementArrange 가 맞는 이름인지 체크 필요 **(완료)**

- AchievementScope, AchievementRange, AchievementApplyTo 등등

AchievementApplyTo 로 적용하였습니다.

3. DBSet 및 Row 이름 정리 하기 **(완료)**

- global 로 뺀것도 빼두기

4. QuestDataChecker 에서 타겟이 스트링이고 목적에 따라 다르게 파싱해야 **(완료)**

var targetString = quest.Value.Target(i);

var targetId = Int32.Parse(targetString);

5. RAchievement 의 오브젝티브는 하나이므로, 단수형으로 바꿔주세요**(완료)**

- RAccumulatedObjectiveProperty 이게 하나밖에 없음

5. RObjectiveProperty 의 타겟 아이디가 없어지고 타겟으로 바뀌였으므로,**(완료)**

오브젝티브타입별 RObjectiveProperty 타입별로 만들어야 합니다.

6. RAchievementSet.Check 에 실제 체크 할 것들 넣어 주세요 **(완료)**

7. 업적은 특성상 매우 많기 때문에, 보상 받지 않은 업적만 쿼리로 가져와서 구성해야 합니다.**(완료)**

-. 다음 진행 스텝은 현재 진행 중인 상태로 문맥을 찾아서 처리해주세요

(Answer). db에서 같은 그룹 내 최고 스탭을 가져와서 완료하였는지를 체크하여

같은 그룹 내 낮은 스텝의 업적이 완료되었다고 판단하는 방법으로 처리하였습니다.

8. 맥락은 메모리를 최소화 해야 함**(완료)**

1번 사항빼고 완료하였습니다.

## 2차

* **(완료)** ConsumableItemType, EquipItemType 의 존재 가치에 대한 고민 필요**(완료)**
  + 이미 소모성 아이템 타입이 분리되어 있는 상태에서 업적용 소모성 타입을 다시 추가하는것이

더 좋은 안전성 또는 생산성에 이점이 있는지 모르겠음

1. ItemCategory에서 소모성 아이템 리스트를 가져와서 itemMainType, itemSubType으로 체크하는 방식으로 변경하였습니다.

* **(완료)**QuestType 이 스토리라인(줄거리)의 종류로 사용되나 Type 자체의 언어 포용성이 너무 커서 혼란을 가중 시킬 수 있어요
  + 이름변경하였습니다.
* **(완료)**오타 수정 : Refeat => Repeat
* **(완료)** FxPlayerItemRemoved 발생 할 때, 데이터 아이디가 있는데, 다시 컨슘에이블 타입을 만들어서 보내는게 중복성이 크다고 생각 되요
* 업적 수락 체크할 때, 보상 받은 그룹 아이디들과 보상 받지 않은 업적 목록 두개로 운용되는데,

**어떻게 하면, 하나로 운용되게 할 수 있는가? 또는 할 수 없다면, 그 이유에 대한 설명이 필요 해요**

* \_rewardAllGroups
  + 사용 형태에 따른 자료 구조 선택에 신경 써주세요

1. **(완료)**insert 부분 체크

return \_nodes

.Node(row.player\_id)

.DoAsync(conn => conn.InsertAndAffectedAsync(row));

2. **(완료)**EquipItemType 제거

3. **(완료)** 2번 적용에 따른 EquipItem All 어떻게 처리 할지

4. **(완료)**Monster Grade 를 넘기지 말고 RActor 데이터 상수로 넘기게 변경

5. **(완료)** RQuestProperty 에서 string questType이 아닌 다른 방식으로 변경

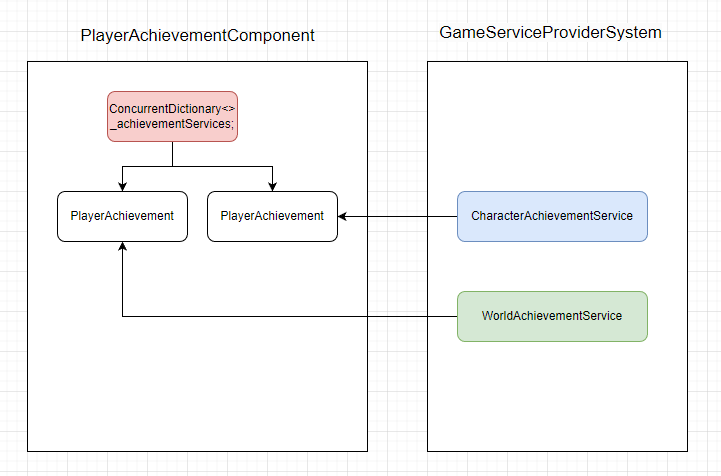
6. **(완료)** 서버에서 캐릭터 생성 후 필수 업적만 서버에서 시작하는 방식으로 변경

* 수정방안) 기획에서 업적 그룹들을 지정하여 그 이후 업적 시작(accept)은 클라이언트에서 시작하게 처리

7. **(완료)** t\_common serialize 누적형 제거

## 3차

1. **(완료)** 업적 캐릭터/월드 저장을 GameServiceProvider 로 빼주세요



2. **(완료)** RAcotr에서는 HashSet을 사용하지 말고 ImmurableHaset을 이용해 주세요

* read only를 하여도 변경에 대한 위험이 있습니다.

의견)

1. **(완료)** 테이블 기법을 활용하여서 복잡함을 해결해 주세요

2. **(완료)** ResourceDummySystem에 UnitTest에 필요한 리소스 작업을 해주세요

3. **(완료)** TryGet~은 실패에 대한 의미가 있습니다 실패가 없는 경우는 Get으로 해주세요

4. **(완료)** 아이템 강화 같이 아이템 관련 정보 변경은 바로 수정하면 안 됩니다. Clone으로 복제 후 작업 해주세요

5. **(완료)** 치트에서 쓰일 “업적 강제 시작”, “업적 전체 지우기” 등도 고려해 주세요

## 4차

1. set에 -1 체크 추가, 음수값 오류 체크

| *// TODO : 랜덤값으로 다시 테스트 필요함*  *// TODO : 음수값일 경우, 어떻게 되는지 확인 필요함* |
| --- |

2. 다이렉트로 db에 사용하지 말고 캐싱을 사용하자.

3. #region을 쓰지 말자

4. 엑셀에 업적 amt 값 0값 오류로 표시되게 추가

* 리소스 체커에 추가

5. 레코딩에 대한 테스트

6. 보상이 최대로 되었는지 추가

## 5차

1. “\_”는 주로 db에서만 사용

2. 다이렉트로 db에 사용하지 말고 캐싱을 사용하자.

## 

## 6차

1. 업적 서비스에서 player/world에 대해 추상화 작업

2. 업적 unit test 작업하였습니다.

* Enhace Prob방식/Assure 방식 혼용 사용에 따른 All 누적치 체크
* 코드 정리하였습니다.

3. quest unit test 추가하였습니다.

* 1번 스텝완료 후, 2/3스텝 완료 시도

## 

## 7차

Q1. 데이터 체크 추가 - PlayTrackType 이 None이 존재하는가?

A2 .업적 데이터 리소스 체커 추가

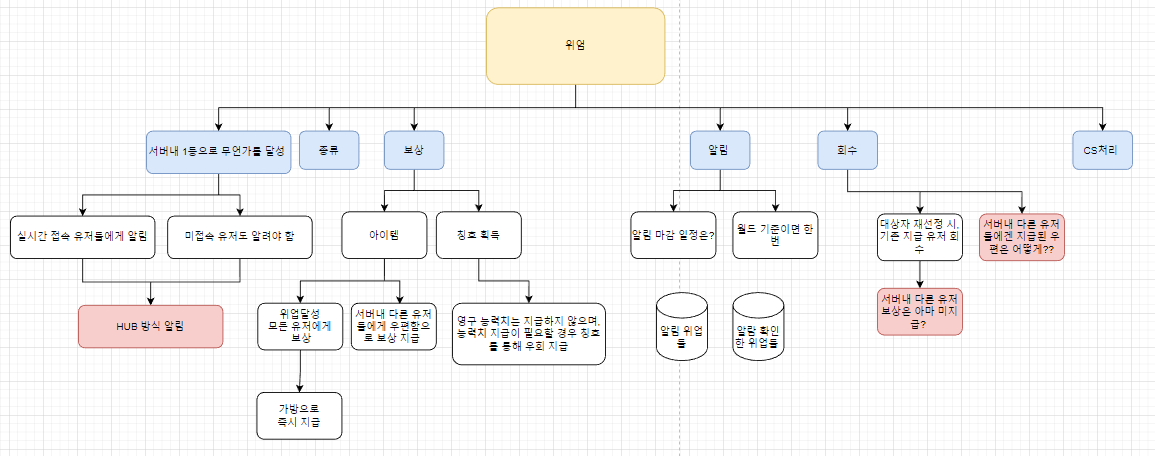
Q2. .CompletedCounts 함수명 변경

(완료) A2. WorldCountsCompleted 로 변경하였습니다.

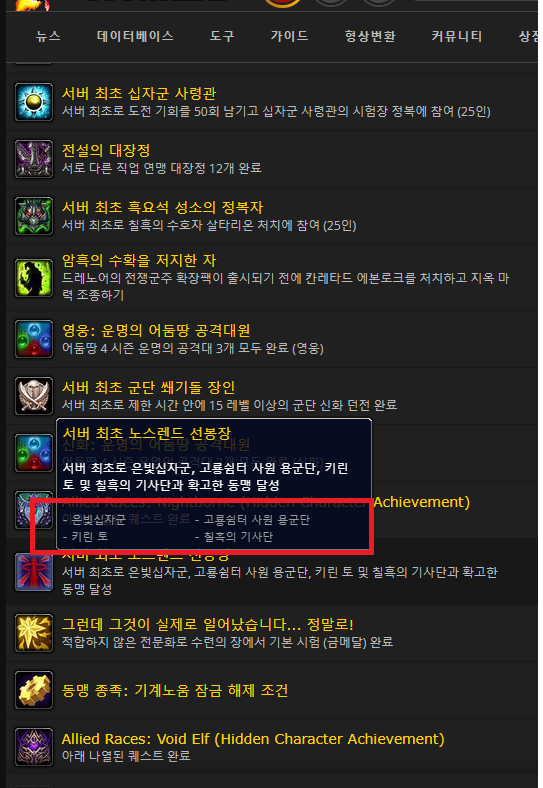
Q3. 업적 보상 unit test에 대해 다시 한번 생각해 주세요.

(완료) A3.

# 위업 작업



## 와우 예시 (위업, 영문명: Feats of Strength)



- xx 던전 최초 완료

- A지역 던전(3개) 완료

- A지역 퀘스트 전체 완료

- A,B,C 업적 완료

- 업적 점수 xx점 이상 획득

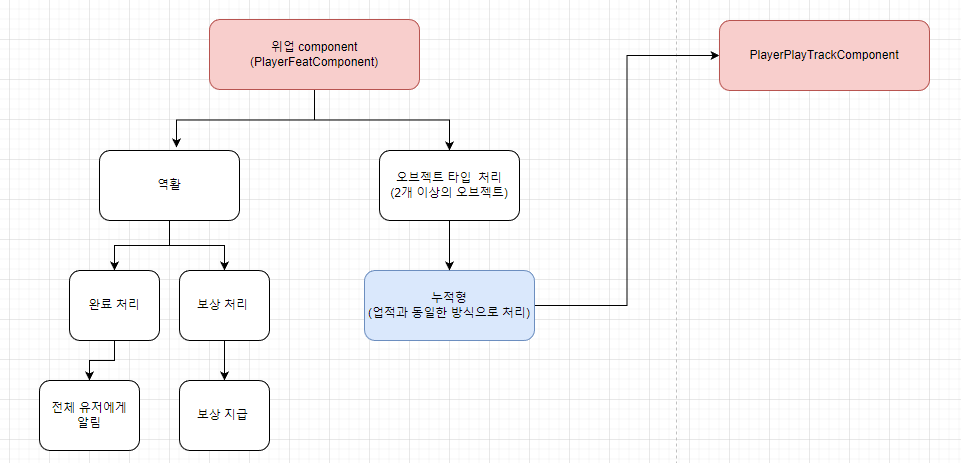
- 최초 x레벨 달성

- xx 세력 확고한 동맹

## 구현

위 예시에 따른 위업 type

| public sealed class FeatType { public static FeatType None { get; }  public static FeatType PlayerLv { get; } // 최초 x레벨 달성 public static FeatType MonsterKillFirst { get; } // 보스 몬스터 최초 사냥 public static FeatType MonsterKillCount { get; } // 몇 마리 이상 사냥 public static FeatType Adventure { get; } // 어느 지역 탐험 public static FeatType QuestCompleted { get; } // 특정 퀘스터들 완료 (예.1001, 1002, 1003 퀘스트 완료) public static FeatType AchievementCompleted { get; } // 특정 업적들 완료 (예.10001, 10002, 10003 업적 완료) public static FeatType AchievementPointOver { get; } // 업적 점수 xx점 이상 획득 } |
| --- |



| Feat = new RFeatSet();  public static RFeat Of(FBDataFeatElement element) |
| --- |

## 알림 처리

