SQL 실행 결과가 Oracle 9i에서는 정렬되어 나타나지만 Oracle 10g이후 버전부터는 정렬된 결과가 나타나지 않는다. 이유를 알아보자

오라클 버전별 SQL 실행 결과 차이: 지금은 왜 정렬되지 않을까?

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	80/12/17	800	(null)	20
2	7499	ALLEN	SALESMAN	7698	81/02/20	1600	300	90
3	7521	WARD	SALESMAN	7698	81/02/22	1250	500	30
4	7566	JONES	MANAGER	7839	81/04/02	2975	(null)	20
5	7654	MARTIN	SALESMAN	7698	81/09/28	1250	1400	90
6	7698	BLAKE	MANAGER	7839	81/05/01	2850	(null)	30
7	7782	CLARK	MANAGER	7839	81/06/09	2450	(null)	10
8	7788	SCOTT	ANALYST	7566	82/12/09	3000	(null)	20
9	7839	KING	PRESIDENT	(null)	81/11/17	5000	(null)	10
10	7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30
11	7876	ADAMS	CLERK	7788	83/01/12	1100	(null)	20
12	7900	JAMES	CLERK	7698	81/12/03	950	(null)	30
13	7902	FORD	ANALYST	7566	81/12/03	3000	(null)	20
14	7934	MILLER	CLERK	7782	82/01/23	1300	(null)	10

SELECT job FROM emp GROUP BY job;

19c(현재버전) Oracle

	JOB
1	CLERK
2	SALESMAN
3	ANALYST
4	MANAGER
5	PRESIDENT

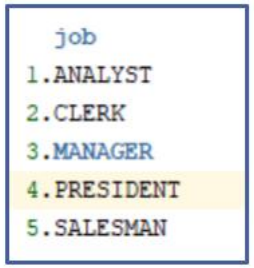
9i(예전버전) Oracle

job
1.ANALYST
2.CLERK
3.MANAGER
4.PRESIDENT
5.SALESMAN

왜 버전마다 다른걸까?

옵티마이저가 처리하는 방식차이

9i	group by => Sort group by join => Nested loop join
10g 이후	group by => hash group by join => hash join / merge join



바로 정렬까지 해준다면
더 좋은거 아닌가?

📊 간단 요약 비교

항목	Sort 방식 (`Sort Group By`, `Sort Merge Join`)	Hash 방식 (`Hash Group By`, `Hash Join`)
초기 안정성	✅ 높음	❌ 낮음 (메모리 제한 등)
성능	❌ 느림 (O(n log n))	✅ 빠름 (O(n))
메모리 의존도	낮음 (디스크 사용)	높음 (메모리 기반)
예측 가능성	높음	낮음 (초기엔 그랬음)
현재 쓰임	조건부로 사용	기본값처럼 자주 사용됨

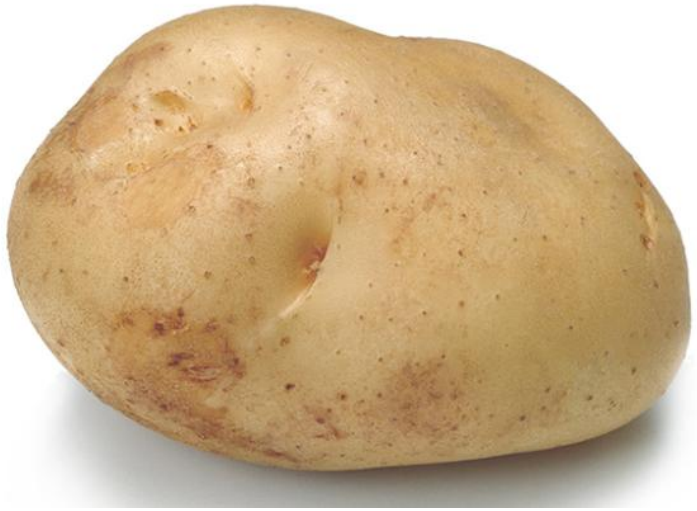
📊 성능 차이 예 (실제 사례 기준)

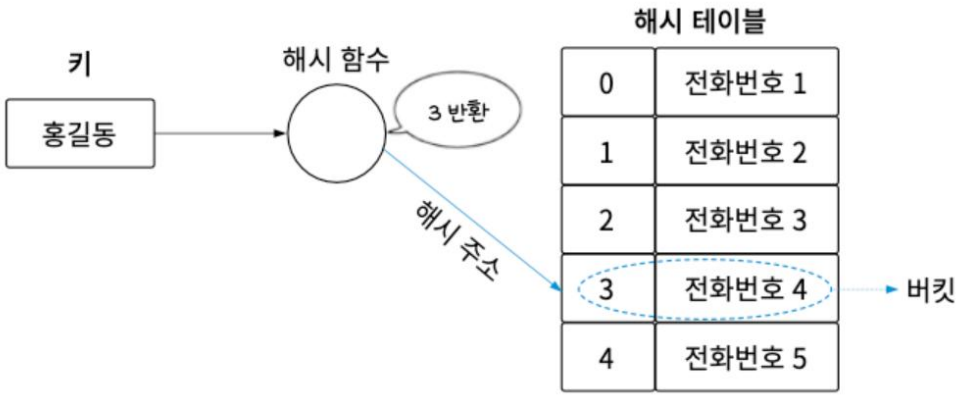
데이터량	Sort Group By 시간	Hash Group By 시간
10만 건	약 1.2초	약 0.3초
100만 건	약 10~15초	약 1~2초
500만 건	수십 초~1분 이상	5~10초 이내

그럼 hash가 뭐길래
이렇게 빠르게 처리해주나

HASH란?

입력값을 고정된 길이의 숫자나 코드로 바꿔
해시테이블에 저장하는 변환 방식





해시를 사용하면 주소록처럼 데이터를 찾아 쓸 수 있어서 좋은 것이군요. 배열은 인덱스를 기억하기 어려우니 일일이 찾아야 했고요.

맞아요. 해시는 해시 함수가 중간에서 값이 있는 위치를 찾는 데 도움을 주니까요. 데이터를 쉽게 찾을 수 있어 좋아요.

해시 함수 (Hash Function): 데이터를 입력받아 해시 값을 계산하는 함수

해시 값 (Hash Value): 해시 함수를 통해 생성된 고정 길이의 값

해시 테이블 (Hash Table): 해시 함수를 사용하여 데이터를 저장하고 검색하는 자료 구조

해싱 (Hashing): 해시 함수를 사용하여 데이터를 해시 테이블에 저장하고 검색하는 과정

📌 해시 테이블의 장점

1. 적은 리소스로 많은 데이터를 효율적으로 관리할 수 있다.
2. 배열의 인덱스(index)를 사용해서 검색, 삽입, 삭제가 빠르다. (위에서 언급한 것처럼 평균 시간복잡도가 O(1) 이다.)
3. 키(key) 와 해시값(Hash)이 연관성이 없어 보안에도 많이 사용된다.
4. 중복을 제거하는데 유용하다.

📌 해시 테이블의 단점

1. 해시 충돌이 발생할 수 있다.
   - 해시 충돌이 발생하는 경우 평균 시간 복잡도가 최악의 경우 O(N) 이 된다.
2. 공간 복잡도가 커진다.◘
   - 대신 시간복잡도가 줄어들었음
3. 순서가 있는 배열에는 어울리지 않는다.
4. 해시 함수 의존도가 높아진다.

출처:https://amazelimi.tistory.com/entry/Python-%ED%95%B4%EC%8B%9CHash%EB%9E%80-%EB%AC%B4%EC%97%87%EC%9D%B8%EA%B0%80feat-Dictionary-%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0

```
diff

ORA_HASH(expr, max_bucket, seed_value)

- expr        : 해시할 대상 (예: job 컬럼)
- max_bucket  : 버킷 최대 번호 (버킷은 0부터 max까지 만들어짐)
- seed_value  : (선택) 해시 결과를 다르게 만들기 위한 시드값
```

Select job, ORA_HASH(job,101,0) AS bucket from emp ;

	JOB	BUCKET
1	CLERK	28
2	SALESMAN	61
3	SALESMAN	61
4	MANAGER	69
5	SALESMAN	61
6	MANAGER	69
7	MANAGER	69
8	ANALYST	37
9	PRESIDENT	75
10	SALESMAN	61
11	CLERK	28
12	CLERK	28
13	ANALYST	37
14	CLERK	28

	JOB
1	CLERK
2	SALESMAN
3	ANALYST
4	MANAGER
5	PRESIDENT

해시함수의 사용 분야

분야	활용 예시	설명
데이터베이스	`Hash Join`, `Hash Group By` 등	대량의 데이터를 빠르게 그룹화하거나 조인할 때 사용. 정렬보다 효율적.
프로그래밍	파이썬 `dict`, `set`, 자바 `HashMap`, `HashSet`	키 기반 빠른 검색, 삽입, 삭제 (O(1)에 가까운 성능)
암호학	SHA-256, SHA-1, MD5 등	비밀번호 등 민감 정보의 단방향 암호화. 복호화 불가.
무결성 검사	파일 체크섬, 해시값 비교	데이터가 전송 중 손상되거나 변조되었는지 확인
캐시 시스템	웹 캐시, CDN, DB 쿼리 결과 캐싱 등	요청을 해시값으로 저장해 빠르게 재사용
블록체인	블록 헤더 해시, 트랜잭션 해시 등	블록 연결성과 위변조 방지를 위한 핵심 기술
중복 제거 / 비교	중복 파일 검색, 데이터 정합성 확인	해시값을 비교해서 중복 여부나 변경 여부 파악