제4부 BI 솔루션

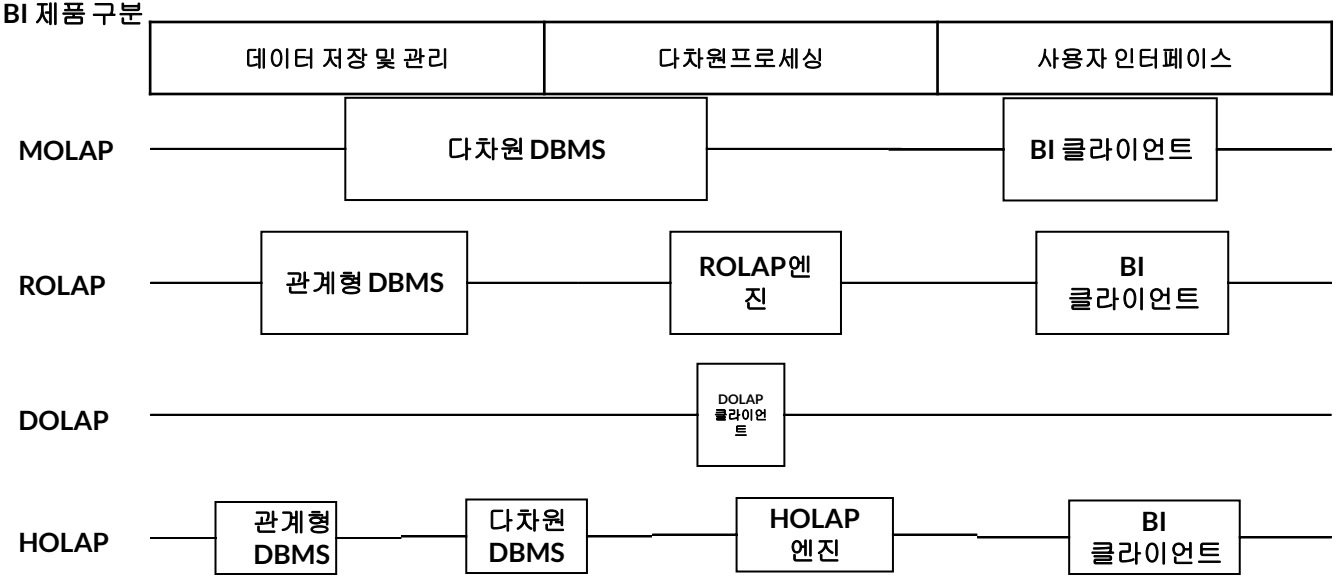
AGENDA

- BI 프로덕트
- BI 표준과 다차원 질의 언어

1. BI시스템 아키텍쳐

- BI시스템 아키텍쳐
  데이터 저장 및 관리 → 다차원 프로세싱 → 사용자 인터페이스(GUI화면) → 사용자

- BI 제품 구분



2. MOLAP

- 다차원 데이터베이스에 기반한 BI 아키텍쳐
- 다차원 데이터의 저장과 프로세싱에 다차원 DBMS가 사용
- 다차원 데이터 저장과 프로세싱에 동일한 엔진이 사용되기 때문에 타 아키텍쳐에 비해 네트워크 상의 데이터 이동이 최소화

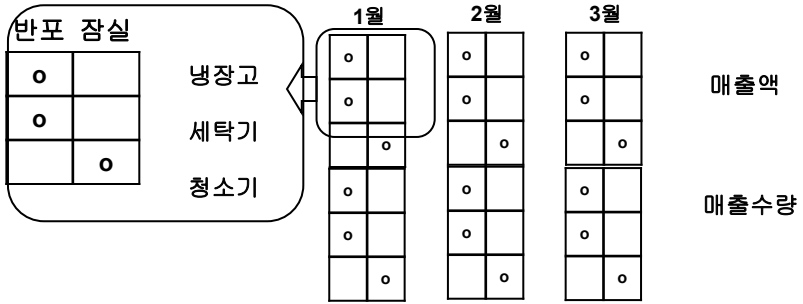
< 다차원 DBMS >

- 복잡한 비즈니스 로직을 쉽게 반영 모델을 구축, 다차원 데이터의 저장과 프로세싱을 효과적으로 수행, 사용자 질의에 빠른 응답
- 신속한 응답 성능을 제공하기위해 다차원 배열 형태의 구조를 사용 (밀집된 형태의 보다 조그만 배열들로 나누어져 저장)
- 관계형 데이터베이스에 비해 데이터 용량, 에러회복 능력, 하드웨어 활용 등의 측면에서 상대적으로 다소 덜어진다.

* 반포매장에서는 청소기를 판매하지 않고
  잠실매장에서는 냉장고와 세탁기를 판매하지않는다고 가정

- 매장차원과 제품차원을 밀집차원으로 기간차원과 변수차원을 희박차원으로 설정할 경우

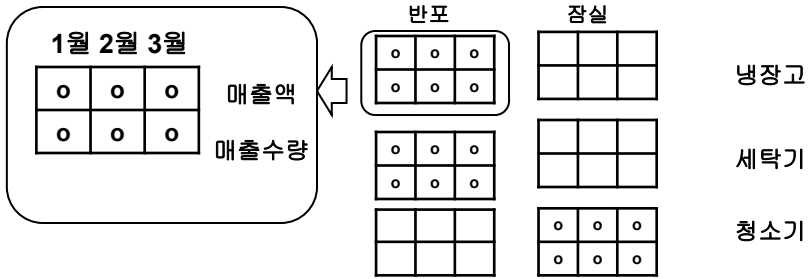
(A)



→ 6개의 블록 생성 각 블록은 6개의 셀로 구성 3개의 셀만 데이터를 가진다. 각 블록은 50%(3/6)의 희박성을 가진다

- 기간차원과 변수차원 밀집차원으로 매장차원과 제품차원을 희박차원으로 설정할 경우

(B)



→ 3개의 블록 생성 각 블록은 6개의 셀로 구성 6개의 셀 모두 데이터를 가진다. 각 블록은 0%(0/6)의 희박성을 가진다

< 배열의 장단점 >

- 배열은 구성하는 하나의 셀 값 만을 필요로 할 경우에도 전체 배열이 메모리에 올라와야 하므로 너무 크지 않게 유지하는 것이 바람직하다
- 함께 자주 요청되는 차원들로 배열을 구성할 경우 한번이 파일 엑세스 만으로 필요한 셀들이 모두 메모리에 읽혀질 수 있으며 최적 성능을 낼 수 있다.
- 배열내의 값은 항목들의 각 조합에 대해 정돈되어 있으며 고정된 위치를 가지므로 인덱스에 영향을 미치지 않고 갱신
- 차원을 구성하는 항목들에 변화가 생길 경우 데이터베이스가 전체적으로 완전히 재구축 되어야 한다.

3. ROLAP

- 다차원 데이터는 관계형 데이터베이스에 저장될 수 있으며 이 경우 스타스키마가 많이 사용된다 (테이블이 물리적 생성된다)
- 질의응답성능향상을 위해 상세테이블과 집계테이블이 만들어질 수 있다.

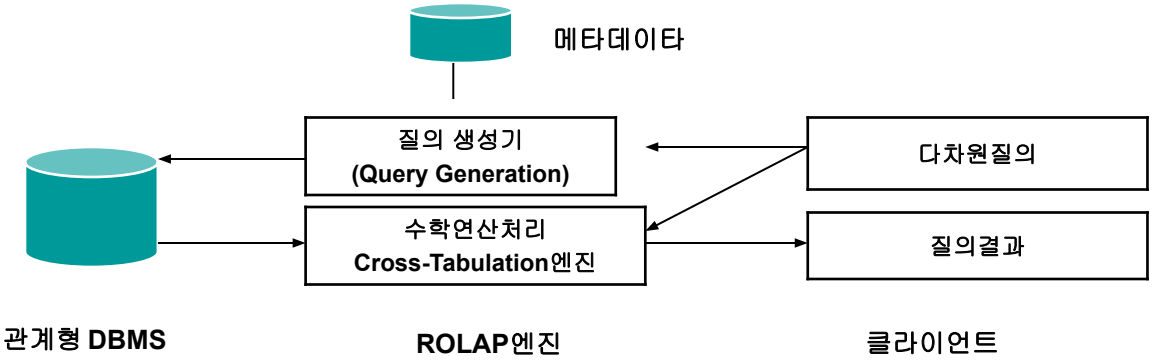
< SQL문의 한계 >

- 분석을 위해 고안되지 않았기 때문에 간단한 질의의 경우에도 SQL문 작성이 매우 힘들거나 불가능하다.
- 비교능력을 결여하고 있다는 점과 순차적 연산을 지원하지 못한다. (이동평균, 상위 x개, 하위x개, 백분율 등)
  ex) 매출액이 가장 좋은 상위 5개 제품은 무엇인가?
- SQL의 한계를 극복하고 보다 쉽게 효과적으로 다차원 질의 문을 작성 할 수 있도록 MDX라는 다차원 질의언어가 개발(MS)

< ROLAP엔진 >

- 사용자와 관계형 DBMS 사이에 위치하여 사용자를 대신해서 복잡한 SQL생성하고 다차원 연산을 수행
- RBI제품들은 메타데이타를 포함해 필요한 모든 데이터를 관계형 데이터베이스에 저장하고 활용하며 다차원 프로세싱을 위해 별도의 BI연산 엔진을 가지고 있다.

- RBI엔진



4. 스타스키마와 관계형 DBMS

- 방대한 데이터를 대상으로 다차원 분석
- 응답 성능에 많은 제한을 가짐

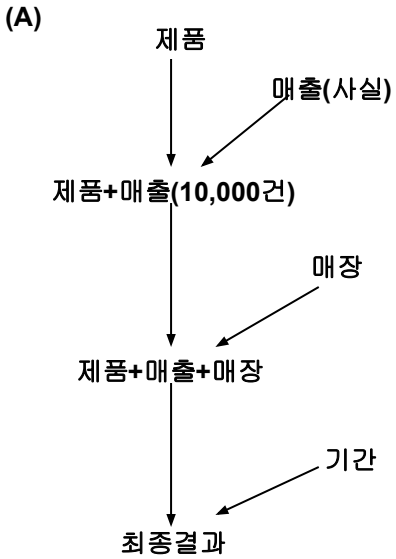
< SQL문의 확장 >

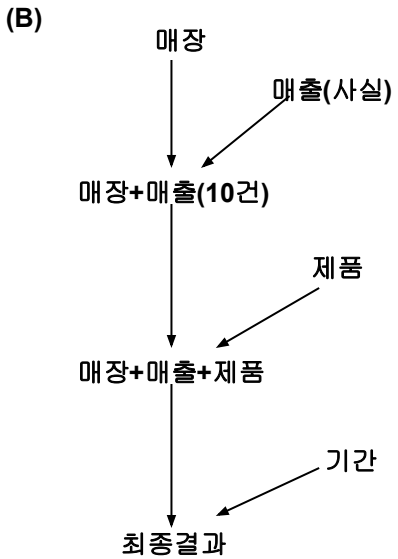
- **RISQL(Red Brick Intelligent SQL)** : 순차적 연산, 문자열 과 수치를 조작, 매크로 구축 할 수 있는 기능 포함

< 스타스키마와 조인 >

* 페어와이즈 조인 : 두개 이상의 테이블을 조인 해야 할 경우 **RDBMS**는 이를 여러 개의 조인으로 분리하여 순차적으로 조인

- 스타스키마와 조인 순서


(A)
제품
매출(사실)
제품+매출(10,000건)
매장
제품+매출+매장
기간
최종결과


(B)
매장
매출(사실)
매장+매출(10건)
제품
매장+매출+제품
기간
최종결과

< 카티션 프로덕트 >

- 테이블 간에 연결되는 칼럼이 없는 경우 조인을 하게 되면 두 테이블이 가진 행들의 모든 조합이 만들어진다.
- 사실테이블과의 조인이 맨 마지막에 이루어져 성능 향상
- 카티션 프로덕트의 크기가 사실테이블의 크기보다 훨씬 작을 경우에만 효과
- 조인문제 궁극적으로 해결 못함. 병렬처리기법과 다양한 인덱싱 기법이 활용됨

5. DOLAP

- 다차원 데이터의 저장 및 프로세싱이 모두 클라이언트에서 이루어진다.
- 분석에 필요한 데이터는 데이터베이스에서 추출되어 클라이언트에 특수한 파일 형태로 저장
- 설치와 관리가 용이하며, 유지보수 부담이 적고 적은 비용으로 **BI**시스템 구축
- 데이터가 모두 클라이언트로 이동될 필요가 존재 이러한 아키텍쳐 문제로 대용량의 데이터 처리에 한계를 가진다.
- 데이터의 일관성을 갖도록 관리하는 문제 발생

6. HOLAP

- 다차원 데이터의 저장공간으로 다차원데이터 베이스와 관계형 데이터베이스가 함께 사용될 수 있는 제품
- 요약된 데이터나 관계식에 의해 새로 계산된 데이터는 다차원 데이터베이스에 저장, 상세 데이터는 **RDBMS**에서 가져온다.
- 빠른 응답성능의 **MBI**의 장점과 확장성이 뛰어난 **RBI**의 장점을 결합한 것

7. BI툴 평가기준

- Codd의 BI제품 평가기준

<기본요소>
1. 다차원 관점 제공 2. 직관적인 데이터 조작 3. 정보 접근성 4. BI데이터의 사전연산 5. 다양한 수준의 BI분석 모델
6. 클라이언트- 서버 아키텍쳐 7. 투명성 8. 다중사용자 지원

<특수요소>
9. 비정규화된 데이터의 처리 10. BI 결과의 분리 저장 11. 널 값의 추출 12. 널 값의 처리

<질의요소>
13. 질의 유연성 14. 질의 응답성능의 일관성 15. 물리적 레벨의 자동 수정

<차원컨트롤>
16. 차원간 동질성 17. 제한 없는 차원 및 레벨 18. 제한 없는 차원 간 조작

- MOLAP과 ROLAP의 비교

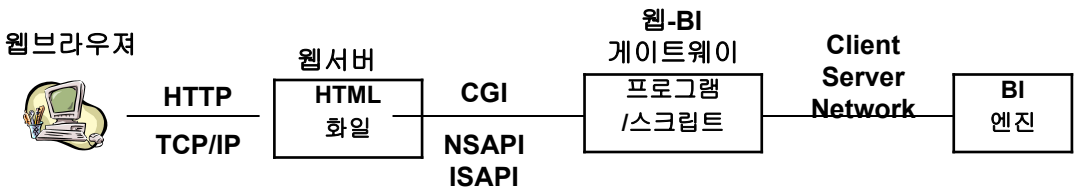
<MOLAP>
1. 많은 연산이 요구되고 연산의 복잡성이 높을수록 2. 행(ROW)사이의 연산이 많을수록
3. 사용자에 의한 데이터베이스 갱신(WHAT-IF 분석)이 필요한 경우 4. 매우 빠른 질의응답성능을 원할때

<ROLAP>
1. 데이터의 희박성이 높을수록 2. 차원항목이나 애트리뷰트가 빈번하게 변경될 경우 3. 많은 차원이 필요한 경우 4. 방대한 데이터를 대상으로 한 경우

8. 웹 BI

- 웹 브라우져를 사용함에 따른 비용절감 (클라이언트 애플리케이션 유지보수, 소프트웨어 설치 및 유지보수)
- 사용자 교육과 관련 비용 절감
- 지리적인 제한 없이 정보에 빠르고 쉽게 접근 (외부 사용자 지원 용이)
- 플랫폼간의 호환성이 좋음
- 인터페이스 방식이나 보안의 문제 발생

-웹 BI 게이트웨이



< 웹 캐스팅 >

- 사용자가 보고서를 미리 등록하고, 설정된 채널을 통해 자동적으로 데이터를 제공 받는다.

BI 표준과 다차원 질의 언어

1. BI 표준

- BI 제품들이 공유할 수 있는 논리적 다차원 모델의 표준이 없음
- BI기술의 확산과 대중화에 걸림돌이 됨
- 표준의 부재는 BI제품들 간에 데이터 교환과 인터페이스를 매우 어렵게 해왔음

< MD-API >

- BI 카운실이 제안한 표준 API 사양 : 실직적인 영향력을 갖지 못함

< OLE DB FOR OLAP >

- 실질적인 BI 표준 API로 받아들여지고 있음.
- 다차원 데이터의 저장환경과 무관하게 다양한 BI 제품들이 용이하게 커뮤니케이션 할 수 있는 기반 제공을 위해 설계
- MDX 라는 다차원 질의언어를 함께 제공

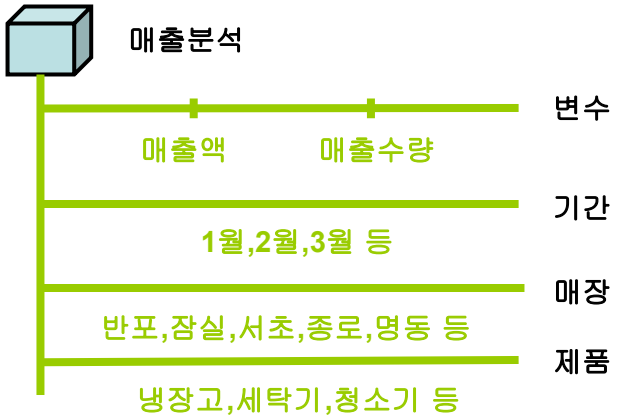
< BI 메타데이터 >

- 표준 API 탄생의 가장 큰 이점은 BI 제품들이 동일한 정보를 공유할 수 있도록 BI메타데이터의 표준이 함께 정립된다는 점이다.
- BI 모델

차원 : 복수의 계층구조를 가질 수 있음
계층구조 : 다수의 레벨을 가질 수 있음
기간과 변수 : 특수한 유형의 차원으로 다른 차원들과 구분
애트리뷰트 : 차원을 구성하는 항목들이 가지는 다양한 특성
OLE DB FOR BI 에서는 레벨에 대해서만 정의 되어진다.

——— MD-API에서만 지원

2. 다차원 질의언어



매출분석보고서

기간차원 ↓

1월		반포	잠실
냉장고	매출액	6,422	5,667
	매출수량	425	390
세탁기	매출액	3,469	4,395
	매출수량	280	312

← 매장차원

제품차원 ↑ 변수차원 ↑

< MDX 구문 >

```
SELECT ([반포],[잠실]) ON COLUMNS,
CROSSJOIN({[냉장고],[세탁기]}, {[매출액],[매출수량]}) ON ROWS  -- 축차원(열과 행을 구성)
FROM [매출분석],[…]    -- 큐브
WHERE ([1월])   -- 슬라이서 차원 (페이지를 구성하는 차원)
```

→ 다차원질의 결과 : **Dataset**

→ 다른 차원에 속하는 항목들이 결합되어 만들어지는 항목들의 모임 : 튜플

(냉장고,매출액),(냉장고,매출수량),(세탁기,매출액),(세탁기,매출수량)

BI 표준과 다차원 질의 언어

< 페이지차원 정의 >
FROM [매출분석]
WHERE ([1월],[매출액],[반포],[세탁기])
→ SELECT 절 없이 모든 차원이 WHERE 절에 명시된 경우 하나의 데이터 값만을 조회
(반포매장의 1월, 세탁기, 매출액 값)

< 항목 선택과 MDX함수 >
→ 셋을 구성하는 항목 지정 방식 : 1. {반포,잠실,서초,종로,명동} 2. {반포:명동}
→ 매장 차원에 속한 모든 항목들로 구성된 셋 생성 : [매장].MEMBERS
→ 강남권의 모든 챠일드 항목들로 구성된 셋을 생성 : [강남권].CHILDREN
→ 조건에 맞는 항목들만 선택
SELECT ([냉장고],[세탁기]) ON COLUNMS,
FILTER ([매장].MEMBERS, [매출액].VALUE > 500) ON ROWS
FROM ….
→ MEMBERS,CHILDREN,FILTER(셋함수) 는 항목이나 튜플, 셋에 적용되어 다릉 셋을 만들어낸다.
→ MDX는 셋함수와 항목함수를 사용해서 모델의 다양한 구성요소와 데이터 값을 활용해 효과적으로 항목을 선택

< 애트리뷰트 조회 >

1월 매출액	담당자	매장평수	냉장고	세탁기
반포	홍길동	12	525	-
잠실	양시향	25	439	930

```
SELECT ([냉장고],[세탁기]) ON COLUMNS,
 ([종로],[잠실]) DIMENSION PROPERTIES [매장명].[담당자],
  [매장명].[매장평수] ON ROWS
  FROM [매출분석]
  WHERE ([매출액],[1월])
```

→
```
SELECT ([냉장고],[세탁기]) ON COLUMNS,
  FILTER([매장].MEMBERS,
  [매장명].[담당자] = '홍길동') ON ROWS
   FROM [매출분석]
   WHERE ([매출액],[1월])
```

BI 표준과 다차원 질의 언어

< 새로운 항목의 계산 >

- MDX와 연산 순서

	2월	3월	월증가율
냉장고	400	500	25
세탁기	200	400	100
주요제품군	600	900	X

```
WITH MEMBER   -- 새로운 항목 정의
  기간.월증가율 =(([3월] -]2월]) / [2월]) * 100 , SOLVE ORDER = 2   --연산의 우선순위 지정
  제품.주요제품군 = 냉장고 + 세탁기, SOLVE ORDER = 1    -- 연산의 우선순위 지정
SELECT ([2월],[3월],[월증가율]) ON COLUMNS,
 (냉장고,세탁기,주요제품군)  ON ROWS
 FROM ….
```

< 희박성과 널(NULL) 값의 처리 >

```
SELECT (냉장고,세탁기) ON COLUMNS,
NON EMPTY {반포:명동}  ON ROWS
FROM ….
```

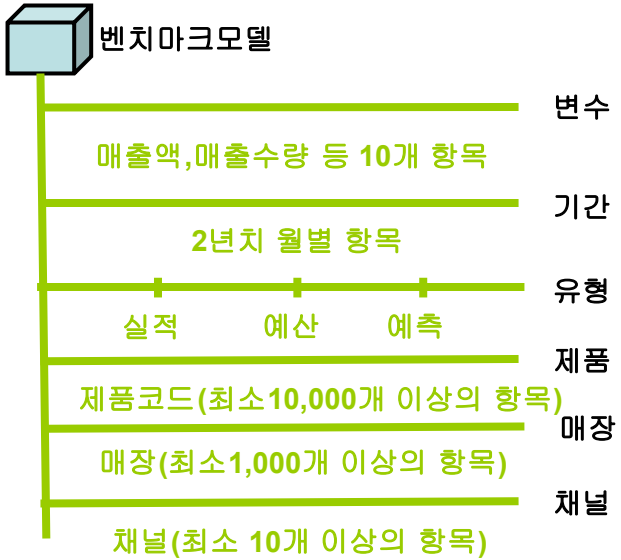
< 기타 MDX구문 >

DRILL-UP , DRILL-DOWN 조작 표현 (DRILLDOWNMEMBER, DRILLUPMEMBER,DRILLDOWNMEMBERTOP ….)

다양한 조건문을 작성하기 위해 IF문과 CASE 문을 제공

3. 벤치마크

- BI 시스템의 전체적인 성능을 측정 : BI 카운실 APB(Application Processing Benchmark)-1
- 방대한 데이터와 사용자를 대상으로 확장, 예측치의 계산방법, 집계방식, 연산순서 등을 명확하게 제시
- 특정 저장매체나 BI 기술에 종속되지 않도록 개발
- 최종결과를 얻기까지 연산에 소요된 모든 시간을 포함하여 성능을 측정



-제품차원 : 전제품,사업부,제품라인,패밀리,제품군,제품류,
제품코드 7개 레벨로 구성된 계층구조를 가짐
- 매장차원: 전체매장, 소매,매장 3개 레벨로 구성
- 채널차원 : 채널계,채널로 구성
- 기간차원 :2년치(1995년,199년)월별 항목으로 구성
분기별,년별로 집계, 월별 년 누계를 가짐
-유형차원 : 실적,예산,예측의 3개 항목으로 구성
실적과 예산은 데이터가 직접 로딩
예측은 실적과 예산 데이터를 기초로 계산
-변수차원 : 매출수량,매출액,재고,표준제조비용,표준운송비용
평균판매가,총비용,수익,수익율,평균매출액

BI 표준과 다차원 질의 언어

< 데이터로딩 >
- APB.EXE 프로그램을 사용하여 벤치마크에 필요한 데이터와 질의를 생성
- 항목과 계층구조를 나타내는 메타데이터와 셀에 로딩될 데이터를 모두 생성
- 메타데이터를 이용해 제품, 매장,채널,기간차원을 구성하는 항목들과 계층구조를 구성
- 셀에 로딩될 데이터는 초기로딩에 사용될 과거 데이터와 주기적갱신에 사용될 현재 데이터로 구분

< 질의 >
- 채널별 매출현황, 매장별 수익성, 재고분석, 시계열 분석, 매장별 예산, 제품별 예산,채널별 예측,
  예산집행 성과, 예측집행 성과 , 애드혹 질의 10가지 유형으로 구분

< 시스템성능 측정 >
- 모든 데이터는 반드시 서버에 저장되고 모든 연산 역시 서버에서 수행될 것을 요구한다
- 벤치마크 수행 과정
  1단계 :APB.EXE 프로그램을 실행하여 항목들의 계층구조를 구성할 파일과 과거 데이터를 생성
  2 단계 : 데이터베이스를 구성하고 과거 데이터를 로딩, 필요시 사전 연산을 수행
  3 단계 : APB.EXE 프로그램을 실행하여 주기적으로 로딩할 데이터 화일을 생성
  4 단계 : 데이터를 로딩하고 필요시 사전연산을 수행
  5 단계 : 질의에 필요한 질의문을 생성하기 위해 APB.EXE 프로그램을 실행
  6단계 : 질의를 수행
- 동시사용자수 최소 10명, 최대 10,000 명을 요구한다,
  각각의 사용자가 수행할 질의 개수는 채널차원을 구성하는 항목 수의 250배 만큼 생성된다
- AQM(Analytical Queries per Minute)측정
  = (실행된 전체 질의 개수 X 60) / 4단계와 6단계를 수행하는데 걸린 총 소요시간(초)
  : 데이터 로딩 및 연산에 소요된 시간을 포함해서 분당 처리된 질의 개수를 나타냄

→ APB-1은 실질적인 BI 환경을 반영할 수 있도록 데이터의 일괄 로딩과 점진적 로딩, 계층구조 상의 데이터 집계,
새로운 데이터의 연산, 시계열분석, 다양한 질의 등과 같은 작업을 수행한다.