1. VCI
   1. 각 데이터의 최대 속도를 활용 -> MBR
   2. 이동 데이터의 최대 속도를 추가적으로 저장한 디스크 기반 R-tree의 변형 색인 구조를 이용
   3. 특정 주기에만 이동 데이터가 자신의 위치를 서버에게 보고 (데이터의 속도가 빨라 특정 주기 내에 위치 추적 불가능한 경우)
   4. 디스크 기반-> 검색시간이 오래 걸림
2. Q-Indexing
   1. 데이터를 색인하는 것이 아닌 질의를 색인하는 질의 색인(Query indexing: Q-indexing) 패러다임이 제안 -> 연속 고정 범위 질의 처리에 적합(유형 3)
   2. 이동 데이터 -> 안전 영역: 데이터의 위치 변화가 영역 내에서 이루어 질 경우 어떠한 질의의 결과에도 영향을 미치지 않음
   3. 서버-이동 데이터 사이의 통신 비용 절감
   4. 안전영역(주로 사이즈가 작음) -> 많은 연산 소모 -> 실시간으로 대량의 질의를 처리해야 하는 실제 상황에 부적합
3. MQM
   1. 분산화된 연산 환경 (서버 부담 감소, 서버 병목현상 완화)
   2. 상주 영역: 이동 데이터가 직접 연산해야 하는 질의를 포함하는 영역 -> 이동 데이터의 연산 능력에 따라 할당
   3. 이동 데이터의 위치가 변할 때 마다 체크할 상황
      1. 상주 영역 밖으로 이동: 새로운 상주 영역 요청
      2. 상주 영역 내 질의 안팎으로 이동: 해당 질의 결과 갱신 요청
   4. 서버와 이동 데이터 사이의 통신비용 절감
   5. BP-tree 구축 시 질의가 중첩되어 저장되므로, 실제 질의 개수보다 많은 수의 질의를 관리
4. MobiEyes
   1. 분산화된 연산 환경
   2. 연속 이동 범위 질의 처리
   3. 그리드 색인 구조 사용
   4. 질의의 범위는 원으로 가정 & 경계 사각형(질의의 중점이 이동할 수 있는 영역)으로 표현, 모니터링 영역(경계 사각형과 겹치는 셀들의 집합)
   5. 이동 데이터의 위치가 변할 시 체크 사항
      1. 모니터링 영역 내: ‘질의 + 속도 벡터’를 서버에게 할당 받고 주기적으로 질의 연산 수행
      2. MQM과 마찬가지로 질의 범위 안팎으로 이동 시에만 질의 결과 갱신 요청
   6. 이동 질의의 위치가 변할 시 서버에게 보고할 경우
      1. 속도 벡터가 크게 변했을 경우
      2. 질의 범위의 중점이 다른 셀로 이동 하였을 경우
   7. 수시로 이동 데이터의 속도가 변하는 상황에서 부적절
5. P2P-MQM
   1. 모바일 P2P 환경 가정, 그리드 기법 사용
   2. 그리드 셀은 이동 데이터의 통신 능력에 의해 설정
   3. 이동 데이터가 속한 셀과 겹치는 질의의 정보를 저장 & 자신의 위치 변화가 질의 결과에 영향을 미칠 시 질의 생성자에서 질의 결과 갱신 요청
   4. 서버가 존재하지 않아 이동 데이터의 연산 능력이 시스템의 성능을 좌우