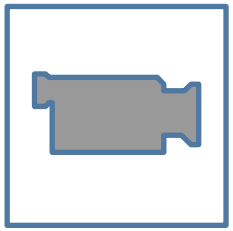
CUBRID 소개 및 발전 계획

작성년월일: 2010년 10월 26일

NHN

목차

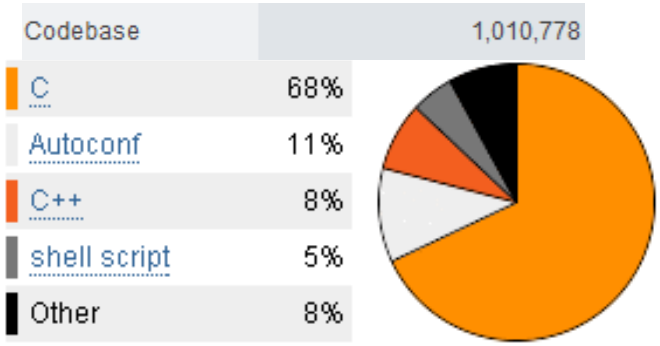
CUBRID 소개



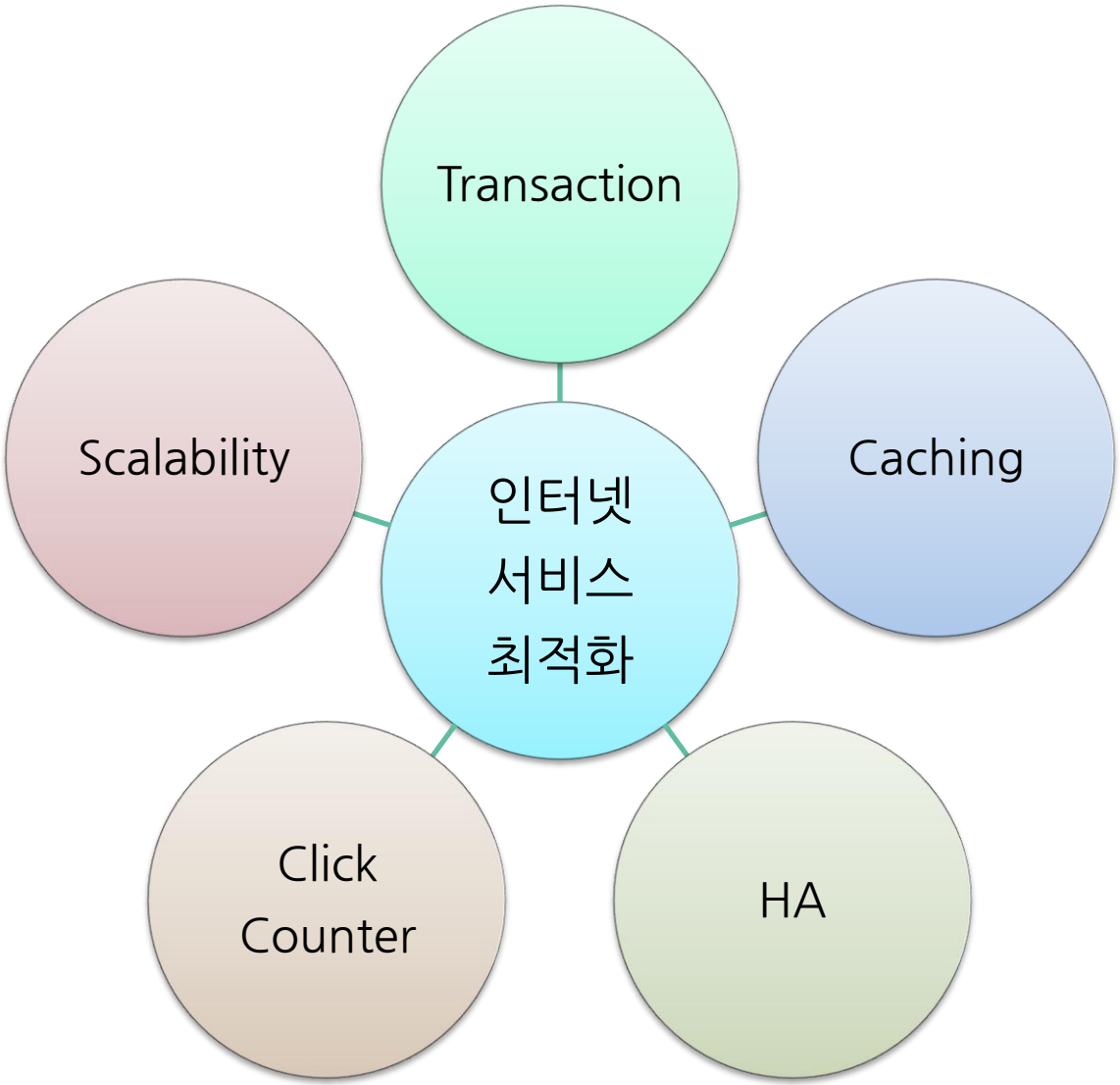
CUBRID...

- CUBRID는
  - “인터넷 서비스에 적합한 DBMS 개발” 프로젝트의 결과물
  - NHN에서 공식적으로 선택 가능한 DBMS 중 하나, 현재 확산 적용중
    - 네이버 핵심 서비스 (카페 덧글과 블로그 덧글)에 적용중

- CUBRID는
  - CUBRID는 2008 R3.0이 2010년 10월 현재 최신 릴리스
  - CUBRID는 1,000,000 라인으로 이루어진
    오픈소스 프로젝트의 산출물



인터넷 서비스를 위한 CUBRID



CUBRID 주요 기능

기능 구분	CUBRID 2008
SQL	SQL-92, SQL-99(ODB)
Data type	Alphanumeric, Large Object (CLOB, BLOB)
Modeling	RDB (Table, column, RI), ODB (Class, method, composition hierarchy, set)
API	JDBC, PHP, ODBC, OLEDB, C API, etc.
Transaction	Record locking, Online backup/recovery
Availability	HA, Replication
Miscellaneous	Partition

- **대용량 RDBMS**
  - 보편성, 확장성, 안정성
  - DB/테이블: 개수 및 크기 무제한
  - 64bit 지원
- **트랜잭션**
  - ACID 보장: commit, rollback, savepoint
  - 다중 단위 잠금: 테이블, 레코드 단위
- **고가용성 기능**
  - 복제, HA
  - 백업 및 복구
    - 온라인/오프라인 백업 지원
    - 증분 백업, 전체백업, 시점 복구
- **다양한 응용 환경**
  - JDBC, PHP, ODBC, OLEDB, Ruby, Python, C API
- **CUBRID Manager**
  - 플랫폼에 독립적인 GUI 개발 및 운영 도구
  - 통합 도구: 관리, 질의, 진단, 튜닝 등

CUBRID 구성

- SERVER
  - 멀티 쓰레드의 고성능 서버 엔진
  - 대용량, 확장성, 가용성을 제공하며, 완벽한 트랜잭션 기능을 제공
  - 고기능, 고성능의 백업/복구 지원

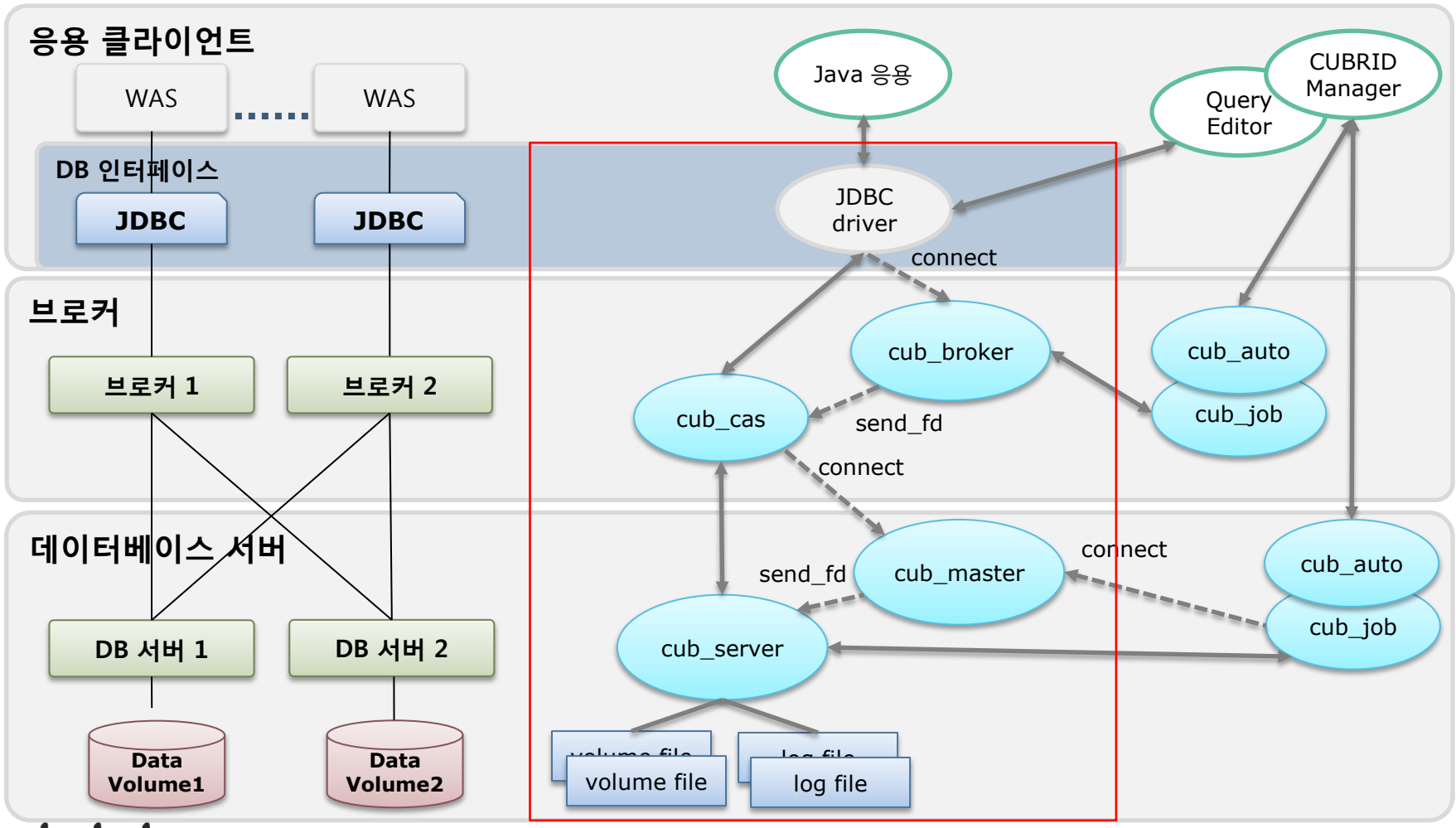
- BROKER
  - 서버와 응용프로그램의 통신을 중계하는 CUBRID의 전용 미들웨어
  - 커넥션 풀링 / 모니터링 / 로그 추적 및 분석 / 로드 밸런스 기능 제공

- 응용 인터페이스
  - 다양한 데이터베이스 연동 인터페이스 제공 (JDBC, PHP, C API, ...)

- MANAGER
  - SERVER 및 BROKER의 모든 기능을 GUI 환경에서 수행하도록 돕는 도구
  - 질의 편집기를 통한 데이터베이스 개발 및 질의를 튜닝할 수 있는 기능 제공
  - SERVER와 BROKER의 운영 현황을 모니터링하고 분석할 수 있는 기능 제공

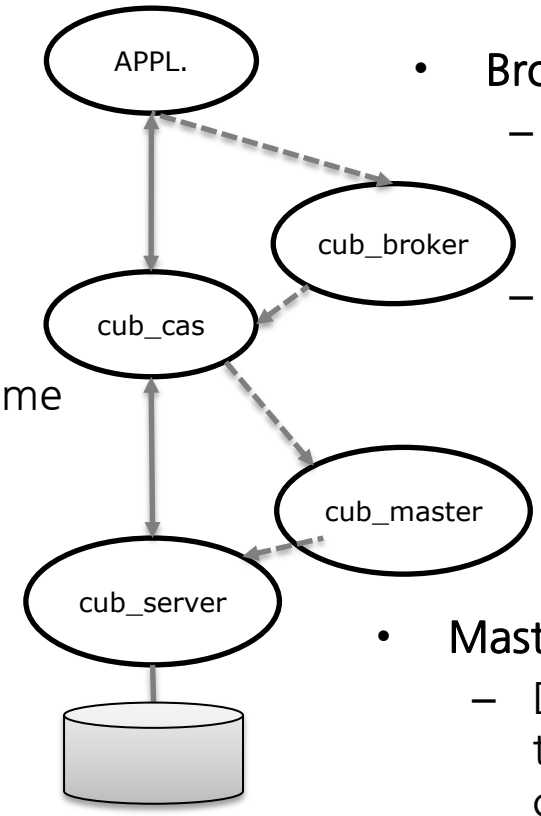
CUBRID 프로세스 구조

- Server와 Broker를 분리하는 3-tier 구조
  - 브로커:DB서버=1:N 가능



CUBRID 프로세스

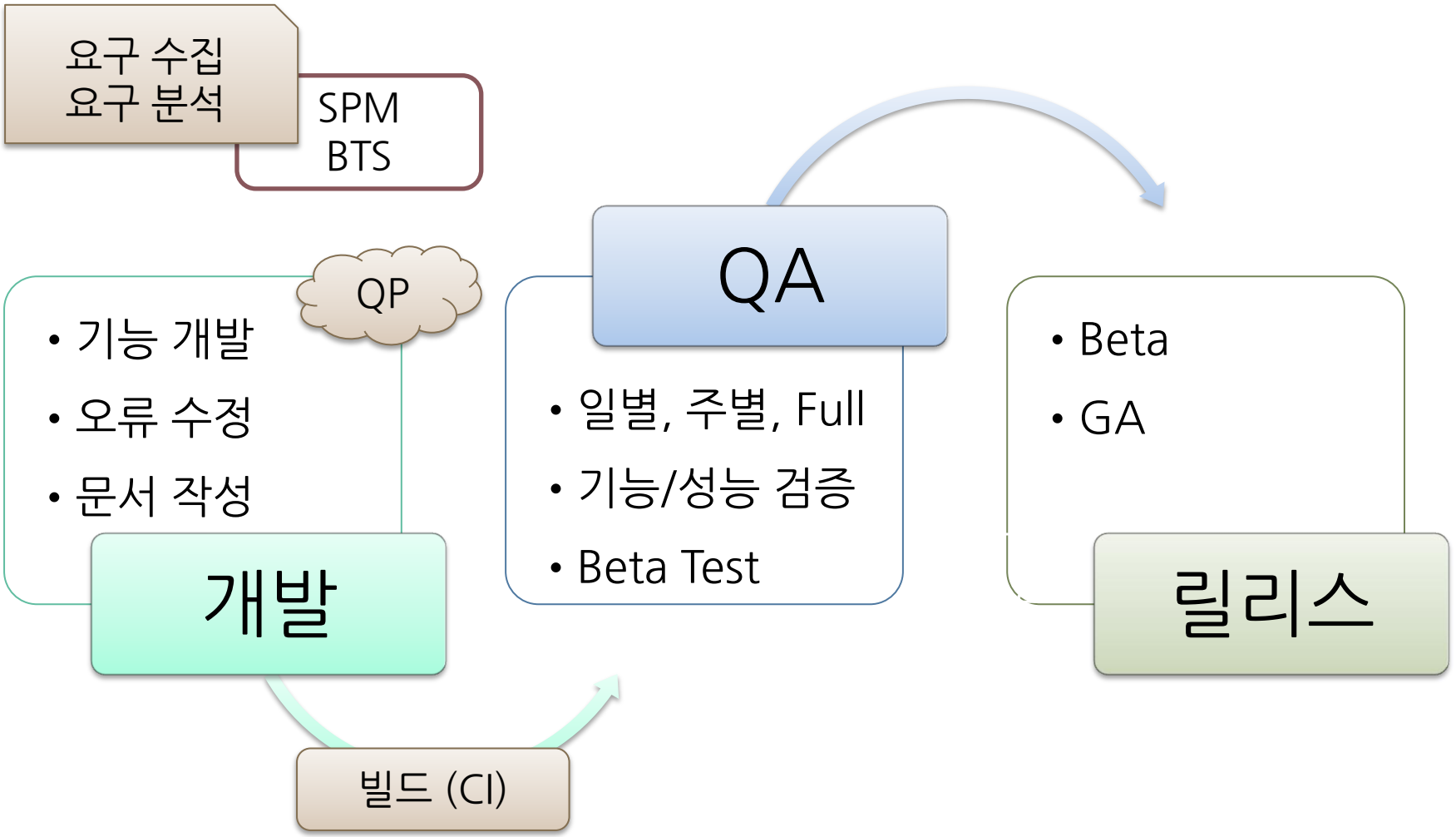
- CAS Process '**cub_cas**'
  - Connects to the server process in behalf of connectors
    - According to the database and user name
- Server Process '**cub_server**'
  - Multi-threaded process serving clients' requests
    - Has thread pool and job queue
  - Accesses database volume and log files
    - One server process per one database



- Broker Process '**cub_broker**'
  - Daemon process listening on the TCP port for connectors' connection
  - Controls cub_cas processes (fork/kill) with connection queue
- Master Process '**cub_master**'
  - Daemon process listening on the TCP port for clients' connection
    - csql, broker, and admin utilities
  - Registrar of server processes using Unix Domain Socket

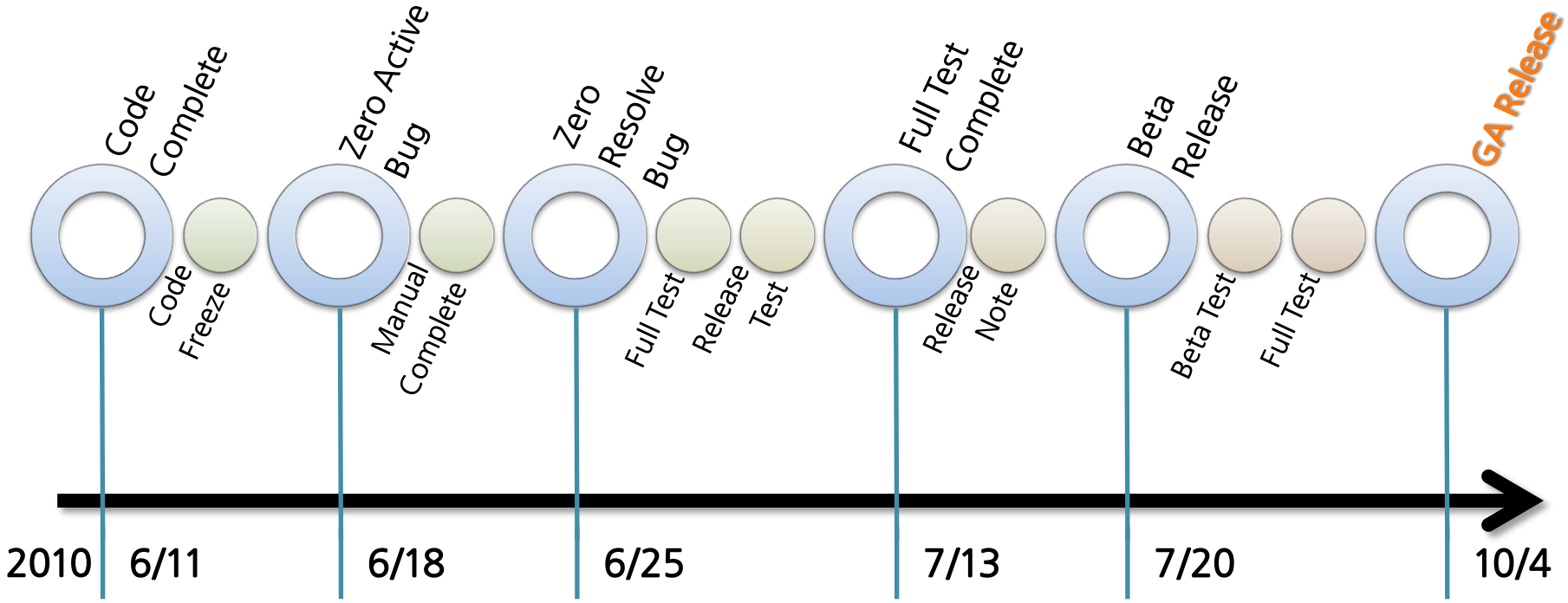
개발 및 적용 현황

CUBRID 개발 과정


요구 수집
요구 분석
SPM
BTS
QP
• 기능 개발
• 오류 수정
• 문서 작성
개발
빌드 (CI)
QA
• 일별, 주별, Full
• 기능/성능 검증
• Beta Test
• Beta
• GA
릴리스

CUBRID 릴리스 과정

예) CUBRID 2008 R3.0 릴리스



CUBRID 개발 현황 - 관련 사이트

http://www.cubrid.com
http://www.cubrid.org
http://cafe.naver.com/studycubrid

CUBRID 적용 현황 - 사내

NAVER 카페 덧글

NAVER 블로그 덧글

NAVER ★ 사전

2010 10.18 오늘의 영화

2010 10.18 오늘의 키친

NHN STORE.

NAVER 개발자 센터

NAVER 공지사항

친절한 나의 스케줄 매니저 네이버 캘린더

NAVER 책으로 자라는 세상

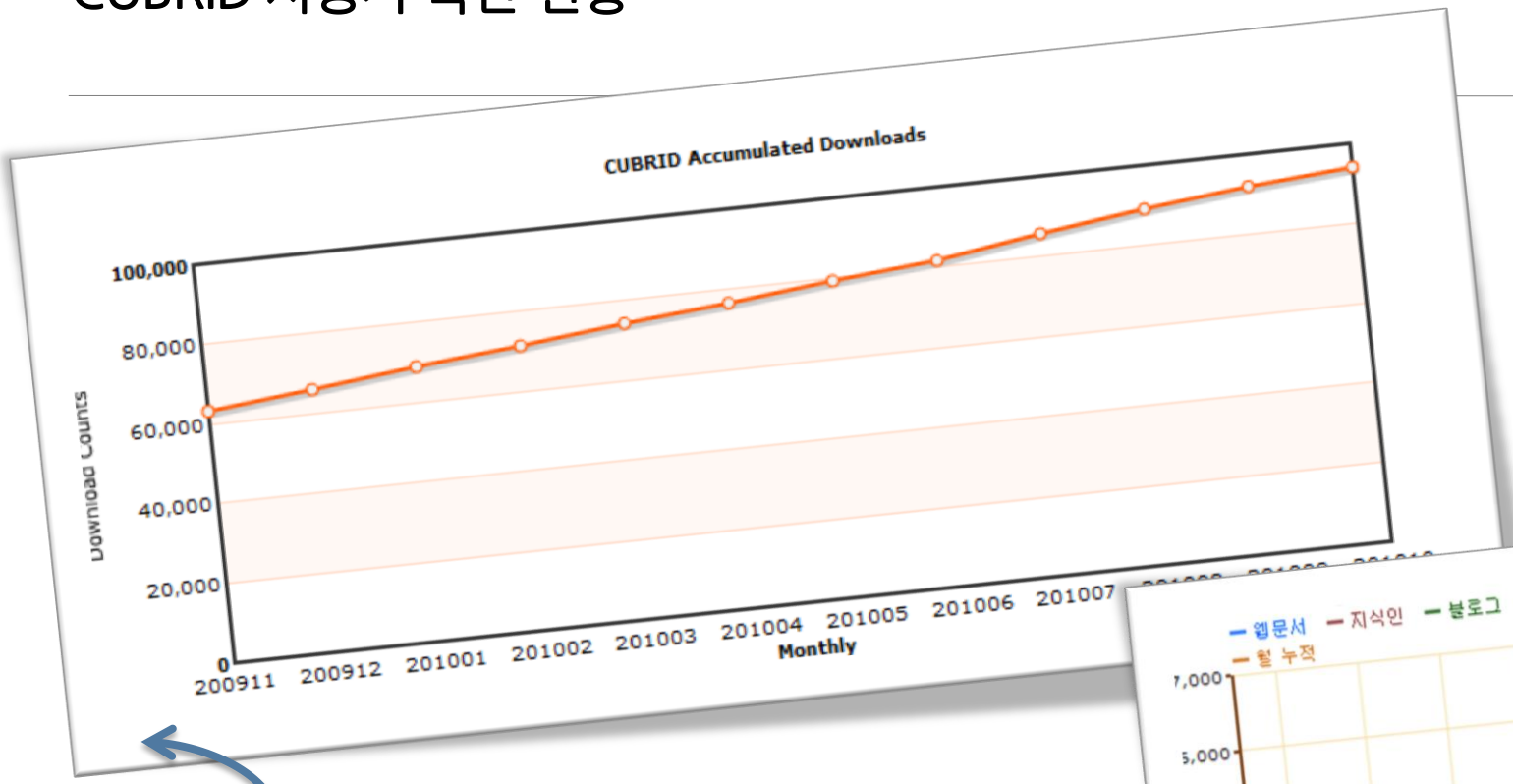
NAVER 웹오피스

NAVER 소셜앱스

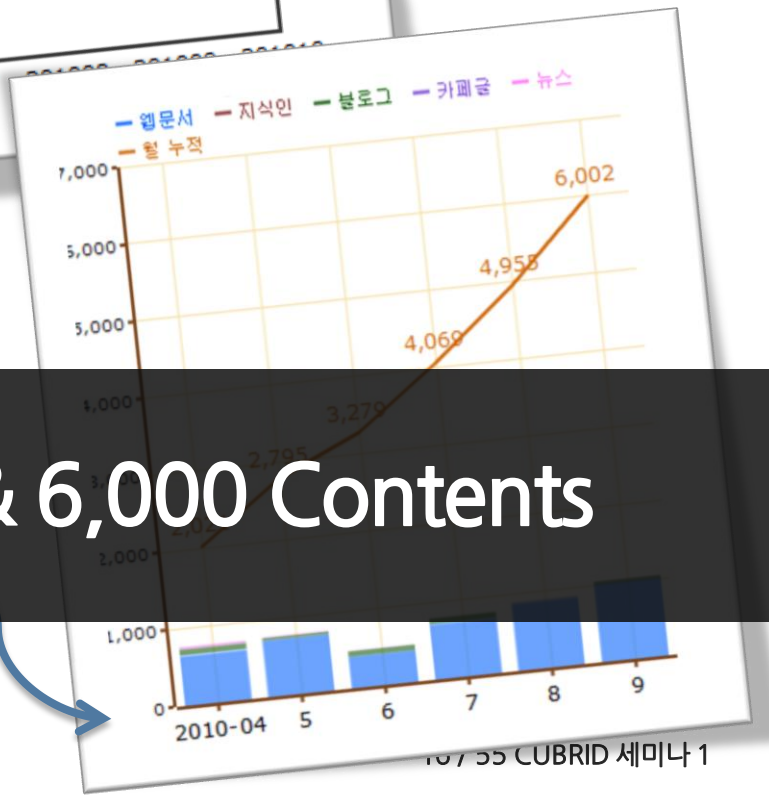
NAVER 단어장

CUBRID 적용 현황 - 사외

CUBRID 사용자 확산 현황



월별 다운로드 추이



월별 컨텐츠 추이

Over 90,000 Downloads & 6,000 Contents

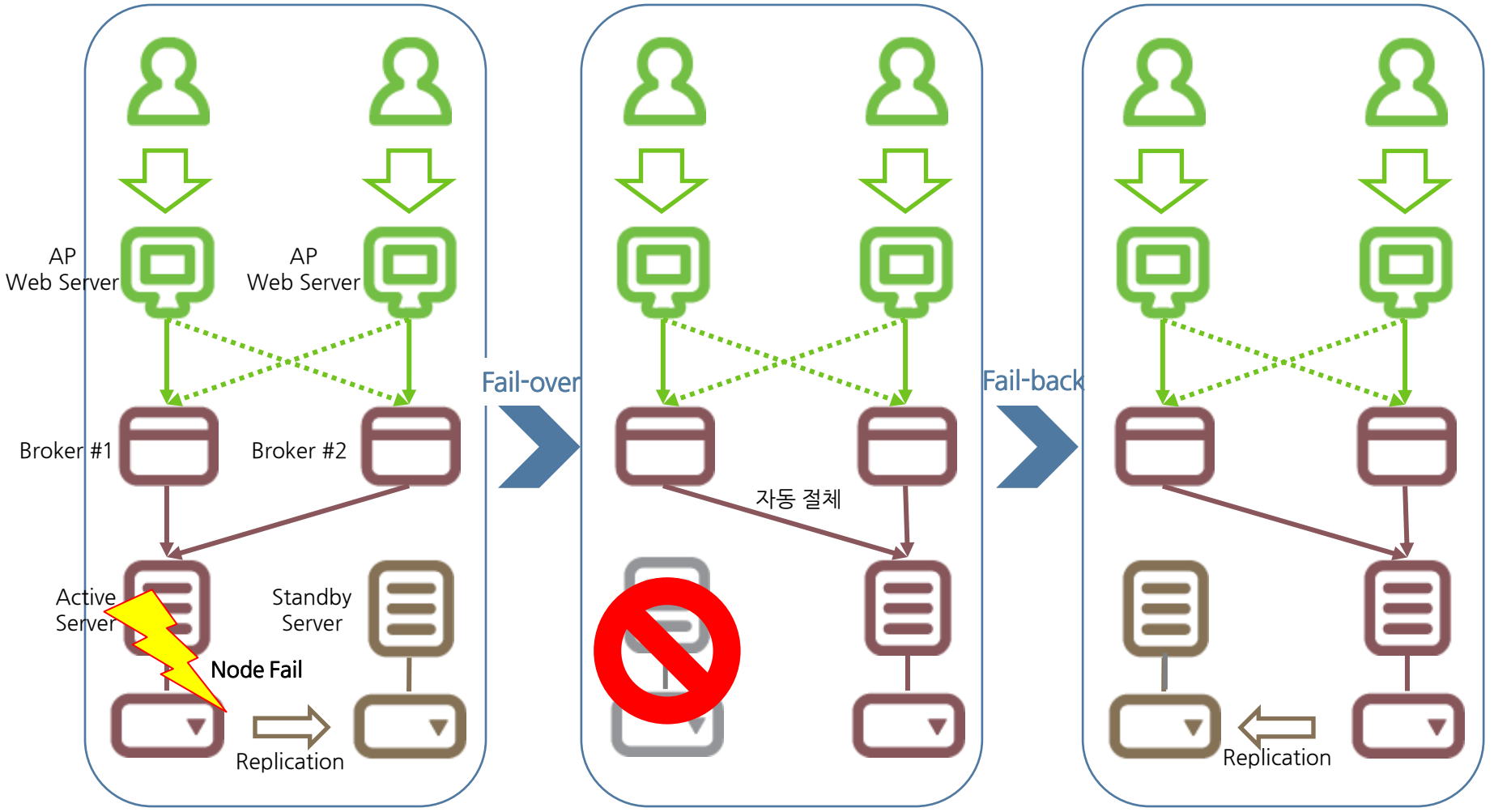
고가용성 기능 소개

CUBRID HA 소개

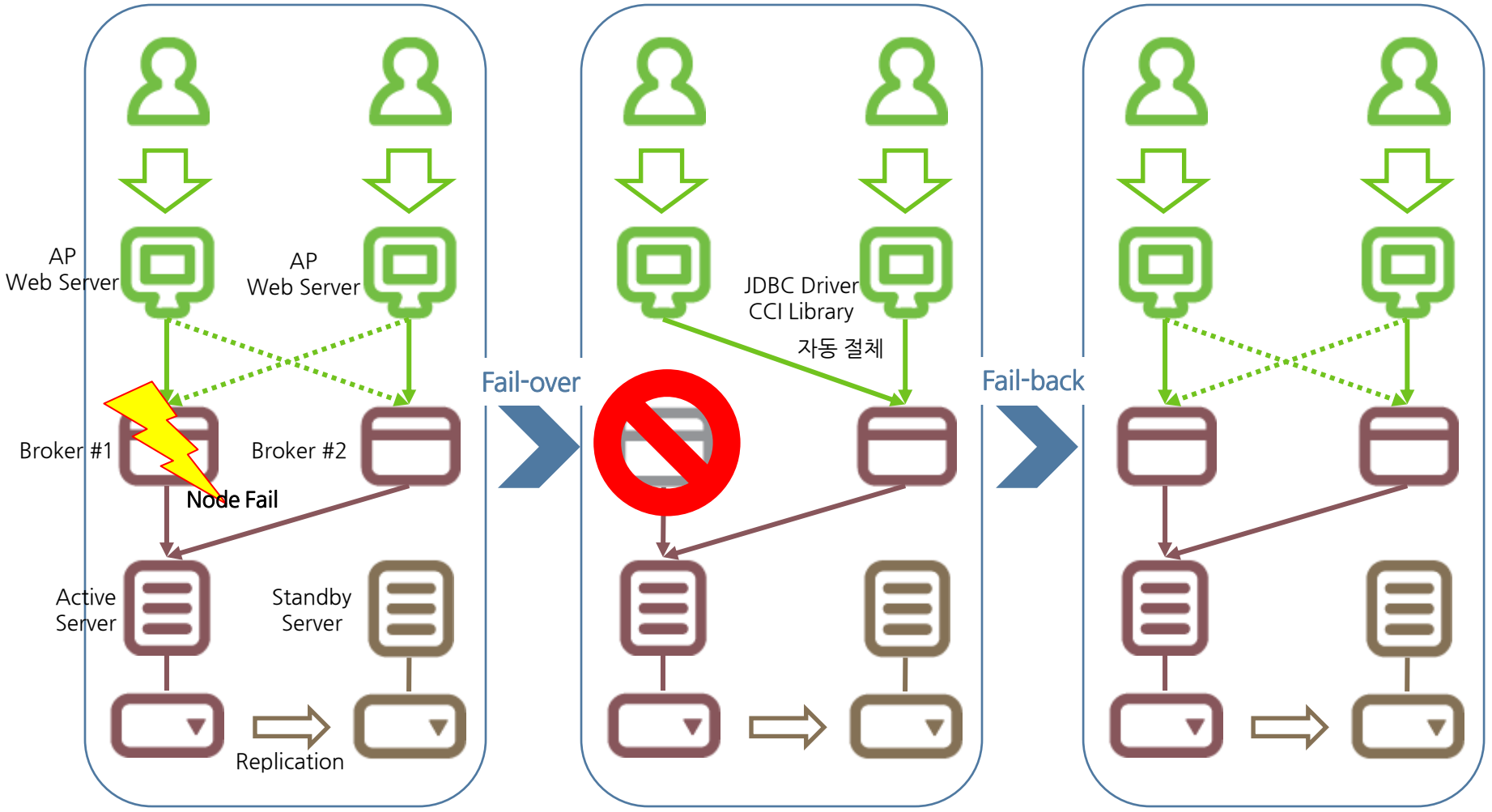
- Shared nothing 구조
- 트랜잭션 로그 기반 복제 사용
- HA mode: sync/semi-sync/async
- 자동 Failover 기능

구분	Shared Nothing	자동 Failover	Failover 시 응용 자동 연결
Oracle RAC	X	O	O
MySQL	O	X*	X*
CUBRID	O	O	O

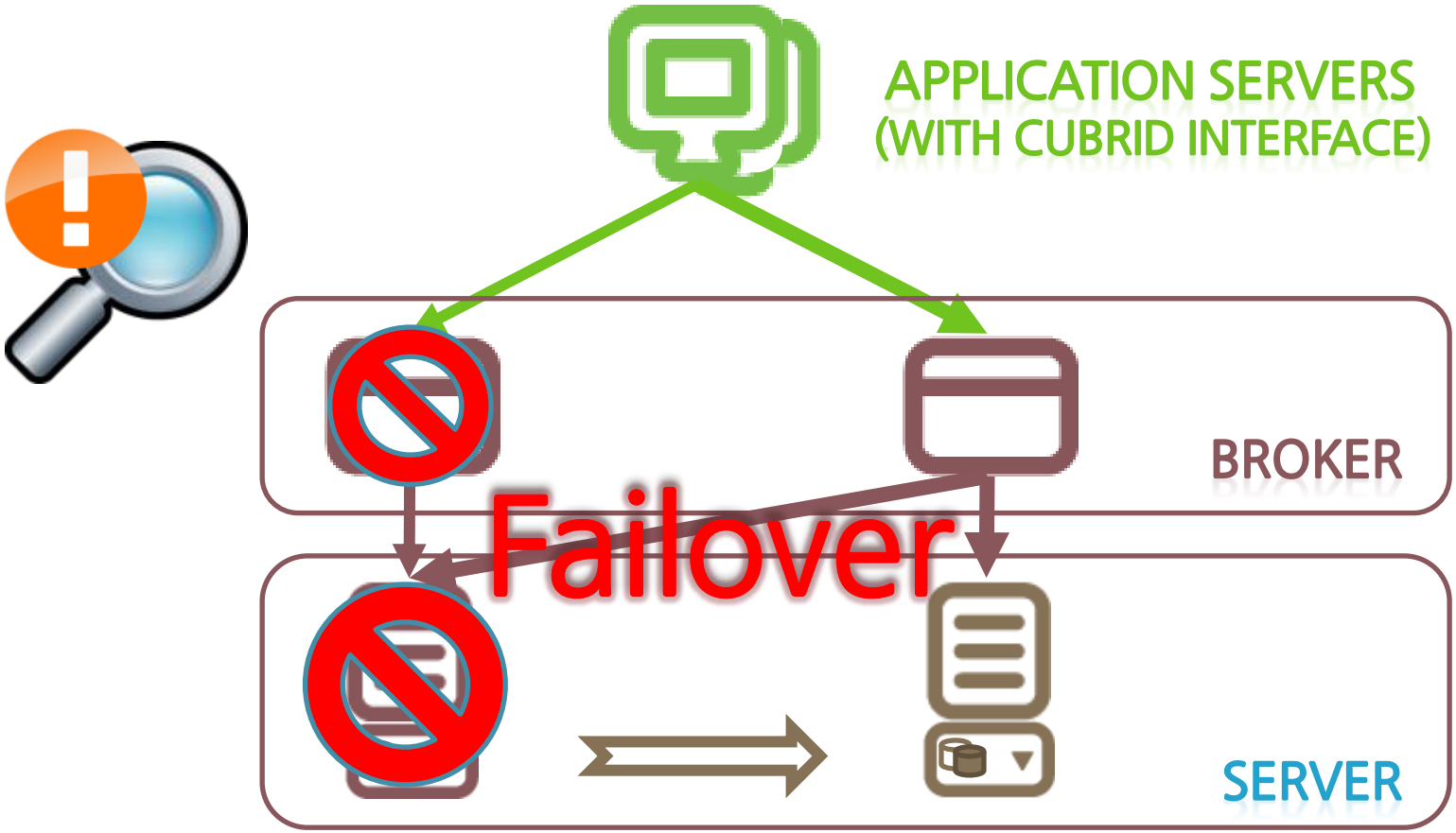
CUBRID HA 구성 - 서버 이중화



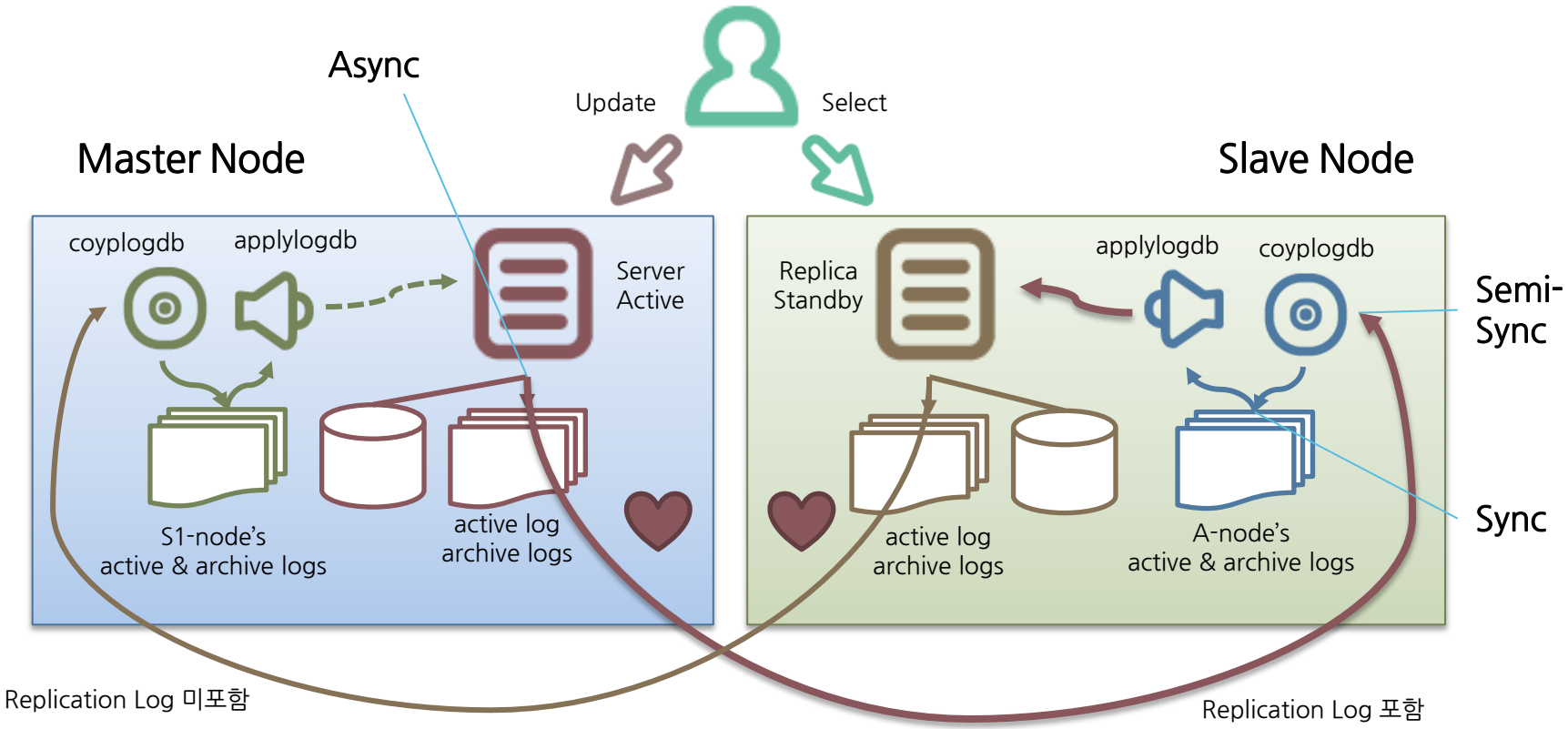
CUBRID HA 구성 - 브로커 이중화



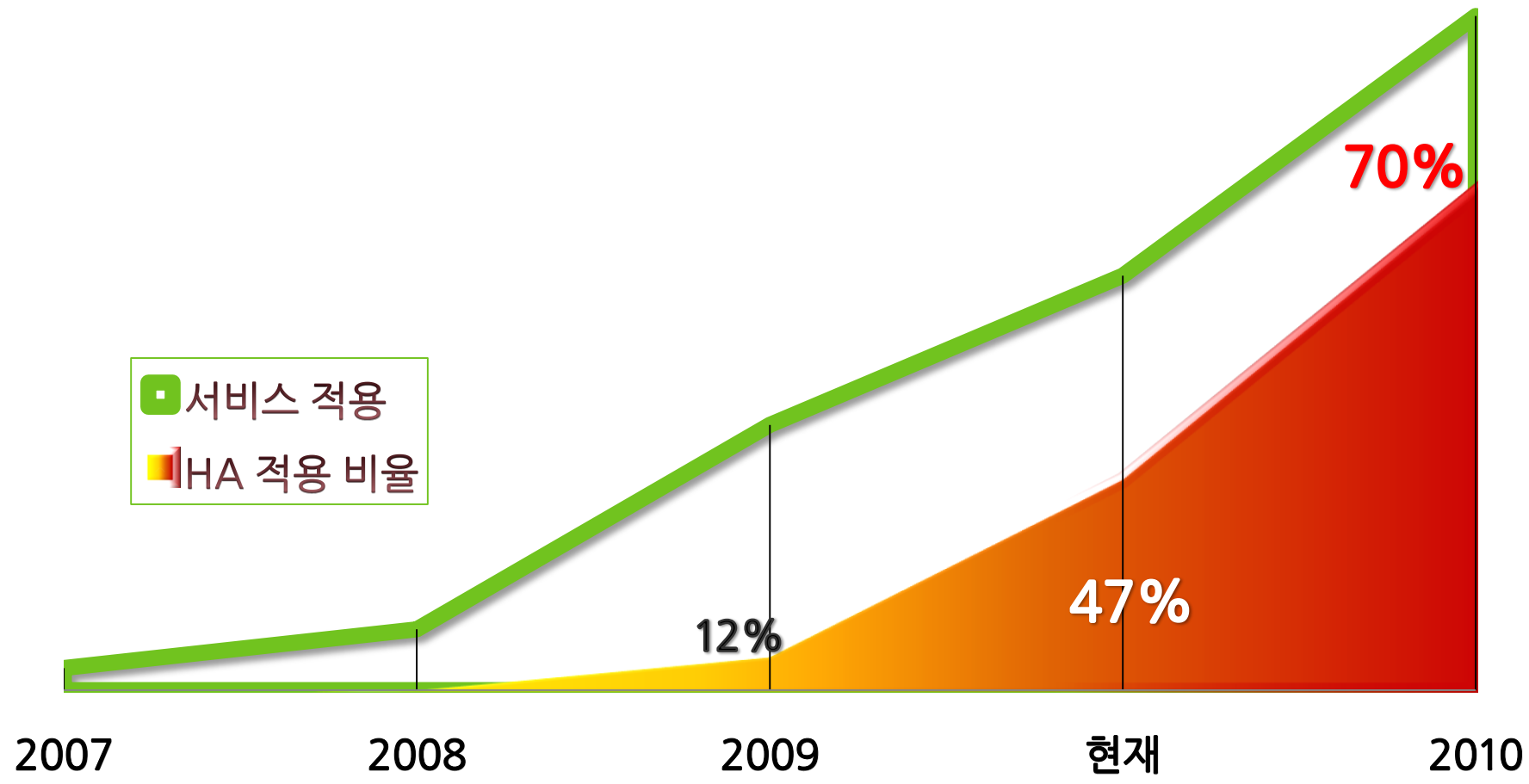
CUBRID HA 동작


APPLICATION SERVERS
(WITH CUBRID INTERFACE)
BROKER
Failover
SERVER

CUBRID HA 동작 - 상세

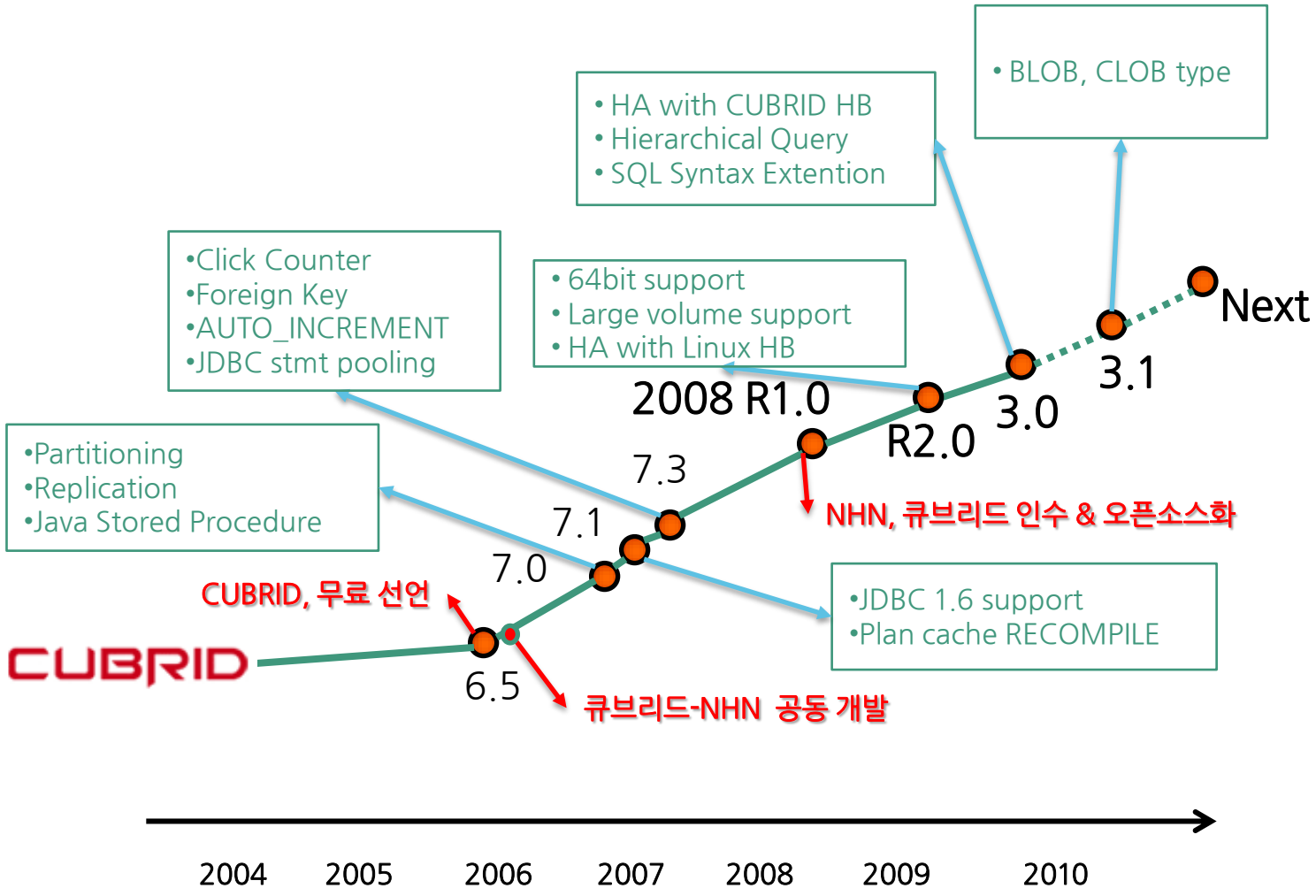


사내 서비스 적용과 HA 적용 비율 현황



릴리스 계획

CUBRID 릴리스 현황



CUBRID 버전별 특징 - 2008 R2.x

- **고가용성 기능**
  - HA 기능 추가 - 2008 R2.0
  - Linux Heartbeat 기능 대체 - 2008 R2.2
    - 관리 편의성, 운영 편의성 향상

- **기능 확장**
  - 64bit 운영체제 지원
  - 2G 이상의 볼륨 사용 가능
  - BIGINT, DATETIME 타입 지원
  - 계층적 질의 (Hierarchical Query) 기능 추가

CUBRID 버전별 특징 - 2008 R2.x

- 성능 개선
  - 데이터 버퍼에 비해 현저히 큰 DB에 대한 랜덤 INSERT 모델 경우 약 400 TPS
    - 부하 모델에 따라 TPS는 차이가 날 수 있음
    - TPS가 0인 구간 제거
- Long query/transaction logging
  - 운영 중 장애를 발생시킬수 있는 slow query 검출 지원
- Manager
  - Eclipse Style UI
  - 다중 호스트 접속 지원, 멀티버전 JDBC 드리이버 지원

CUBRID 버전별 특징 - 2008 R3.0

- **SQL 확장, 호환성 향상** - MySQL 지원 구문 중심
  - CREATE 문 확장: 기존 테이블의 스키마, 데이터를 이용한 테이블 생성
  - INSERT 문 확장: 멀티 레코드 입력 가능, REPLACE 문 지원
  - SELECT 문 확장: FROM 절 생략, LIMIT 절 지원
  - ALTER 문 확장: 추가할 컬럼의 위치 지정
  - 그 외, TRUNCATE, DO 문 등 다수 지원

  - 응용 개발 편의성 및 이식성 향상 기대
  - 구문이 MySQL과 동일한 경우에도 의미는 다를 수 있음
    - 하위 호환성 유지
    - 추가 확장 계획

CUBRID 버전별 특징 - 2008 R3.0

- **연산자 및 함수 추가** - 약 80개
  - 논리 연산자, 비교 연산자, 비트 연산자 추가 지원
  - 수학 함수, 날짜/시간 데이터 함수 추가
  - 다양한 날짜 출력 포맷 지정자 지원
  - 다양한 문자열 함수, 정보 함수를 추가 지원

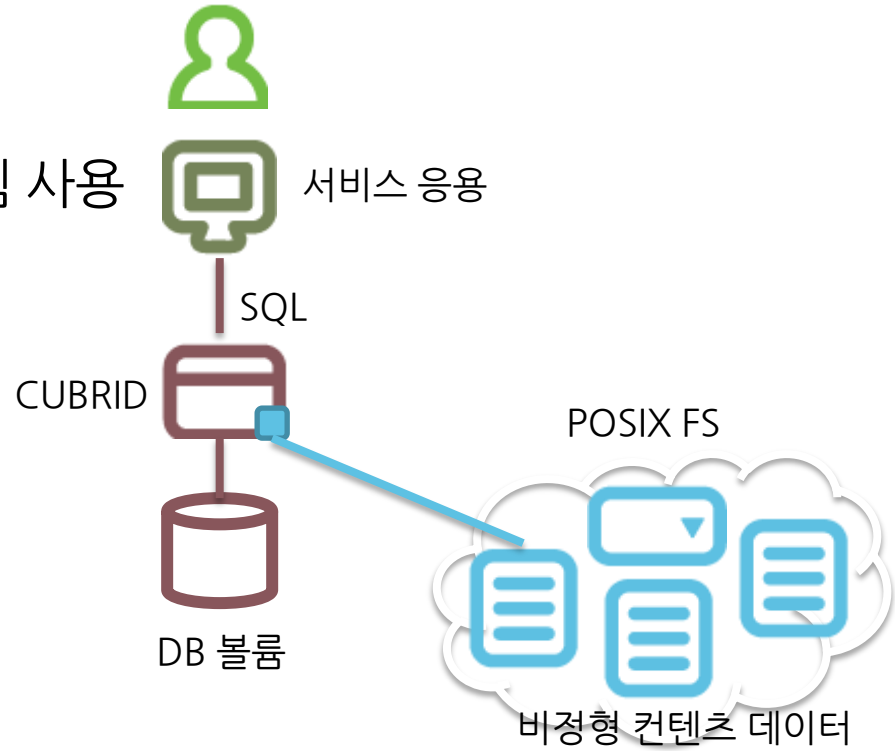
- **운영 편의성 및 안정성**
  - CUBRID HA 기능 안정화
  - 데이터베이스 공간 정리 작업을 온라인에서 수행하는 옵션 지원
  - 베타 테스트 기간 동안 새로운 검증 시나리오 추가 수행 및 수정

CUBRID 버전별 특징 - 2008 R3.1

- **개발 배경**
  - 컨텐츠(비정형 데이터)는 파일 시스템에 저장하고 메타 정보(정형 데이터)는 DB에 저장하는 유형이 많다
  - DB 접근 후 파일 시스템에 접근을 하는 2단계 작업을 하고 있다

  - 응용이 DB 인터페이스 만으로 파일 시스템에 저장된 컨텐츠를 한번에 access할 수 있는 기능을 제공하자
  - 대용량 컨텐츠, 첨부 파일 등에 대한 공간 확장성, 안정성 제공하자

- CUBRID **분산파일시스템**
  - BLOB/CLOB data type 지원
  - DB에는 저장 위치를 기록
  - FBO 데이터를 POSIX 파일시스템 사용



관련 질의 예)

CREATE TABLE doc (doc_id VARCHAR(10), *content CLOB*)

INSERT INTO doc VALUES ('doc-1', *CHAR_TO_CLOB* ('very~long ~text'))

CUBRID Coming up next

- 성능 향상
  - INDEX Enhancement
  - 저장 구조 개선
- SQL 확장 2단계
- HA 기능 안정성 및 편의성 개선

CUBRID Coming up next - KEYLIMIT

- LIMIT predicate은 최종 결과를 제한
  - 하지만, 최종 결과는 적지만 INDEX 스캔 중간 결과가 매우 많다면?
    - 스캔 범위가 넓은 경우
    - non-unique index에서 매칭되는 키 값을 가지는 레코드가 많은 경우
    - 해당 데이터를 읽지는 않더라도 키 값에 매칭되는 OID를 찾기 위한 인덱스 페이지 접근은 최소한 반드시 필요
  - ➢ 성능 문제 있을 수 있다

CUBRID Coming up next - KEYLIMIT

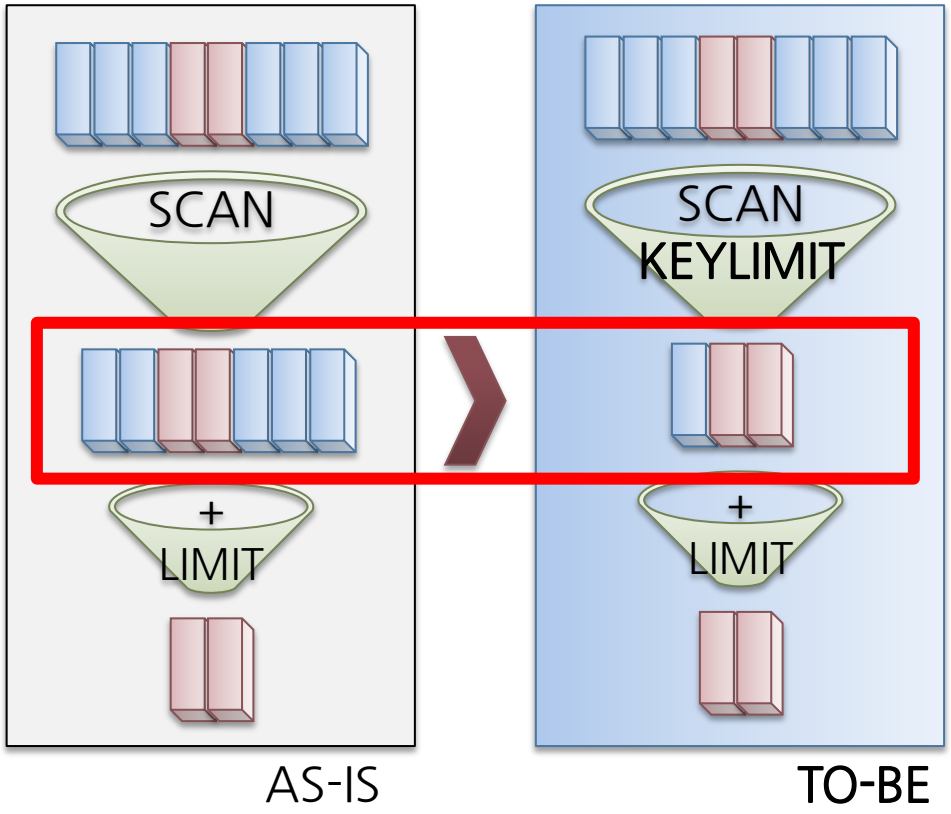
- WHERE predicate
  - Key Range
    - INDEX 스캔 범위로 활용되는 조건
    - (a BETWEEN 1 and 100)
  - Key Filter
    - Key Range에 포함될 수 없지만 INDEX 키로 처리 가능한 조건
    - (a <> 50)
  - Data Filter
    - INDEX를 통해서 처리될 수 없는 나머지 조건
    - (c < 10)

CUBRID Coming up next - KEYLIMIT

- CUBRID의 질의 처리 과정
  - INDEX 스캔인 경우, 먼저 Key Range와 Key Filter로 INDEX 스캔 수행
  - INDEX 스캔 결과(조건을 만족하는 OID 집합)에 대해서 레코드를 읽어서 Data Filter 적용
  - ➢ LIMIT를 통해서 최종 결과를 제한할 수 있지만, 중간 결과를 제한할 수는 없다

CUBRID Coming up next - KEYLIMIT

- Syntax & Semantics
  - USING INDEX [table.]index[*KEYLIMIT [from,] N*]
  - INDEX 스캔 과정에서 원하는 N개를 찾으면 스캔을 중단하고 리턴
  - 일종의 힌트
- Data Filter 없는 경우
  - LIMIT 절을 KEYLIMIT 절로 자동 rewrite optimize



CUBRID Coming up next - LIMIT 최적화

- Rewrite optimize
  - Data Filter 없는 경우 KEYLIMIT로 처리
  - orderby_num을 ROWNUM으로 처리

- Top K sorting
  - 정렬을 해야 하는 경우에도 전체를 정렬한 결과를 생성한 후에 K개를 취하기 보다는 top K sorting을 통해서 최적화

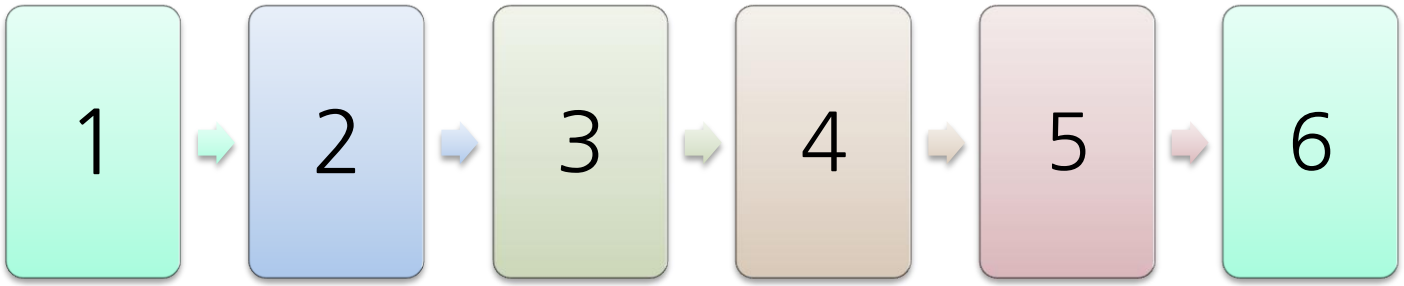
CUBRID Coming up next - Pagination

- **전형적인 예**
  - SELECT * FROM forum WHERE ...
    ORDER BY docno desc LIMIT ?, ?;

- **문제**
  - 글이 많으면 뒤 쪽 페이지로 이동할수록 느려진다
  - 왜?
    - 러시아 페인트공 문제 

- INSERT로 인한 문제
  - 다음 페이지로 이동했는데 이전 페이지에서 봤던 내용이 또 다시 출력된다
  - 사용자들이 진정 원하는 것이 이런 패턴일까?
  - 모바일 환경에서는 바뀌어야 하지 않을까?
  - 이런 패턴은 최적화하기 매우 어려움

- CUBRID가 생각하는 해법
  - 특정 페이지로 점프하여 탐색할 수 있다면 빨리 검색할 수 있다
  - 마치 스캔을 stop-resume하는 개념
  - 마지막 위치 이전에 새로 입력된 것은 후속 질의 결과에 영향을 주지 않는다



CUBRID Coming up next - Pagination

- 기존 SQL syntax/semantics로 가능한가?
  - 처음 N개를 가져오는 질의와 이후 M개를 계속 탐색하는 구문 필요
    - SELECT INDX_CRT_POS([*table.*]*index_name, N*)
      INTO :*last_pos*;
    - SELECT ..., INDX_CRT_POS() FROM ... STARTING
      AT :*last_pos* NEXT *M* ...;

- 1단계 최적화에서 고려중인 제약사항
  - Primary Key, Unique Index만 적용 가능
  - 한 테이블에 대해서만 가능

※ 아직 확정된 것이 아님

CUBRID Coming up next - INDEX 개선 on ordering/grouping

- Use multicolumn index on ordering and grouping
  - SELECT * FROM t1 ORDER BY a, b, c;
  - SELECT * FROM t1 WHERE a=1 GROUP BY b,c
- Use index full scan for ordering
  - SELECT * FROM t1 WHERE b < 100 ORDER BY a;
- Descending access
  - SELECT * FROM t1 ORDER BY a DESC;
- Move operation in LHS to RHS and use index
  - SELECT * FROM t1 WHERE b - 1 > 0 ORDER BY b;

CUBRID Coming up next - LIKE 최적화

- LIKE predicate
  - 기본적으로 full scan으로 처리됨
  - 지금까지 예외적으로 INDEX scan이 가능했던 경우
    - range로 변환 가능한 경우
    - s LIKE ‘blah’+‘%’
    - ➢ s >= ‘blah’ and s < ‘blai’

CUBRID Coming up next - LIKE 최적화

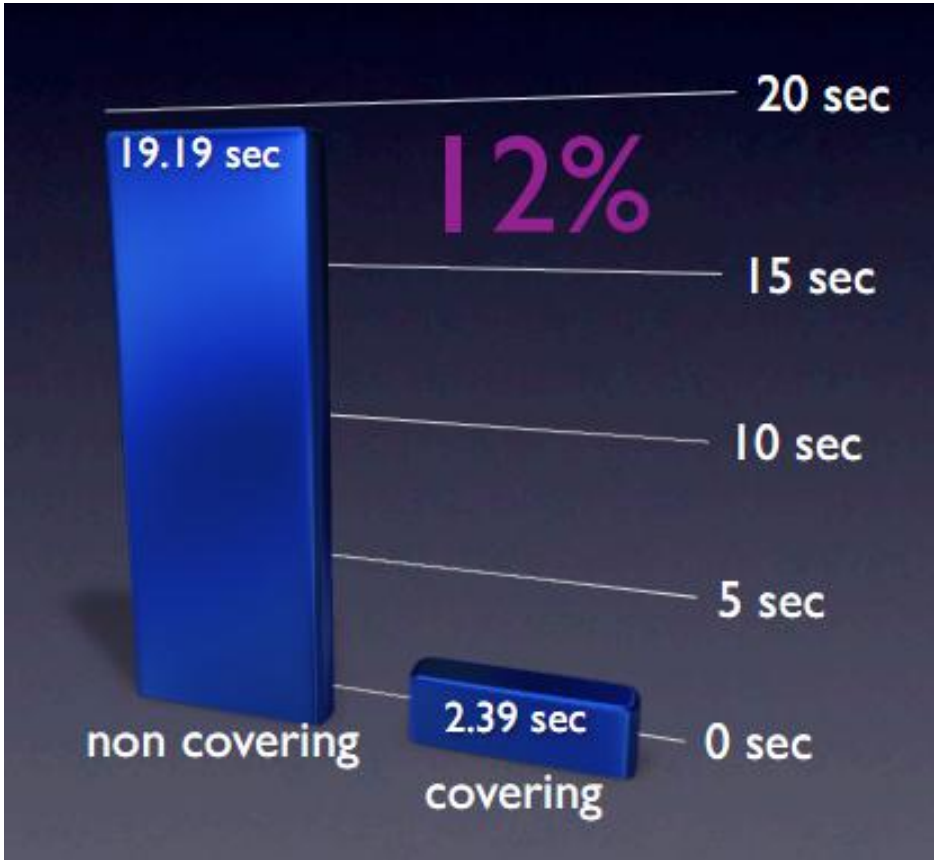
- Rewrite optimization
  - s LIKE ?+‘%’;
  - ➢ s >= ?:1 and s < ?:2 and s LIKE ?:0 + ‘%’;
- Full index scan
  - s LIKE ?+‘%’
  - s LIKE ‘%’+?+‘%’
  - s LIKE ?
  - ➢ full index scan하면서 LIKE 조건을 key filter로 적용

CUBRID Coming up next - Covering Index

- **INDEX scan 성능 향상을 위해**서
  - 질의 내에서 언급되는 컬럼이 모두 INDEX 키에 포함되어 있다면, 데이터 페이지에 대한 접근 없이
    INDEX 접근만으로 질의 결과 생성
  - INDEX 페이지의 버퍼링 효과 높임
  - 때로는 INDEX 필터링 조건에 포함되지 않는 컬럼을 INDEX에 포함 시키는 튜닝 필요

CUBRID Coming up next - Covering Index

- 시험
  - Total records: 1억 7280만건
  - 약 20,000++건의 임의 데이터 조회
  - cold start w/o OS cache



CUBRID Coming up next - 저장 구조 개선

- **INDEX 페이지 사용량 줄이기**
  - 키 저장 구조 변경
  - 기존 대비 약 50% 정도 용량 감소. 성능 개선 효과 큼
- **데이터 페이지 사용량 줄이기**
  - variable length 저장 등 overhead 최소화
  - 기존 대비 약 20% 정도 감소
- **Slotted 페이지 구조 변경**
  - slot 구조 등 overhead 최소화

➢ Disk I/O 사용량 최소화 → 성능 향상과 직결

CUBRID Coming up next - 저장 구조 개선

CUBRID Coming up next - 저장 구조 개선

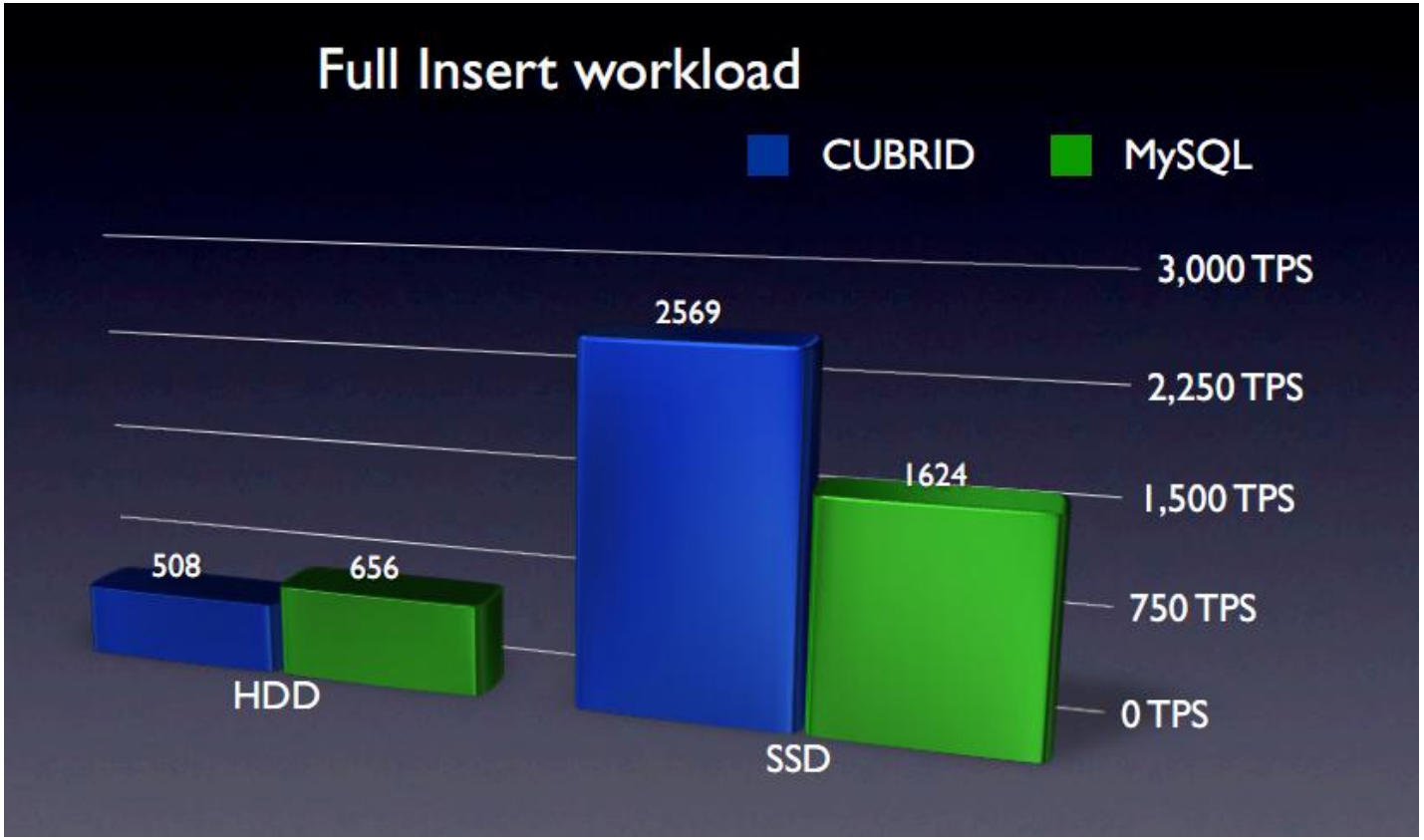
CUBRID Coming up next - SQL 확장, HA 개선

- SQL 확장 2단계
  - Implicit type casting
  - Host variable binding
  - 구문, 함수 추가 지원, 그외 다수

- HA 개선
  - 기능 및 성능 안정성
  - 운영 편의성

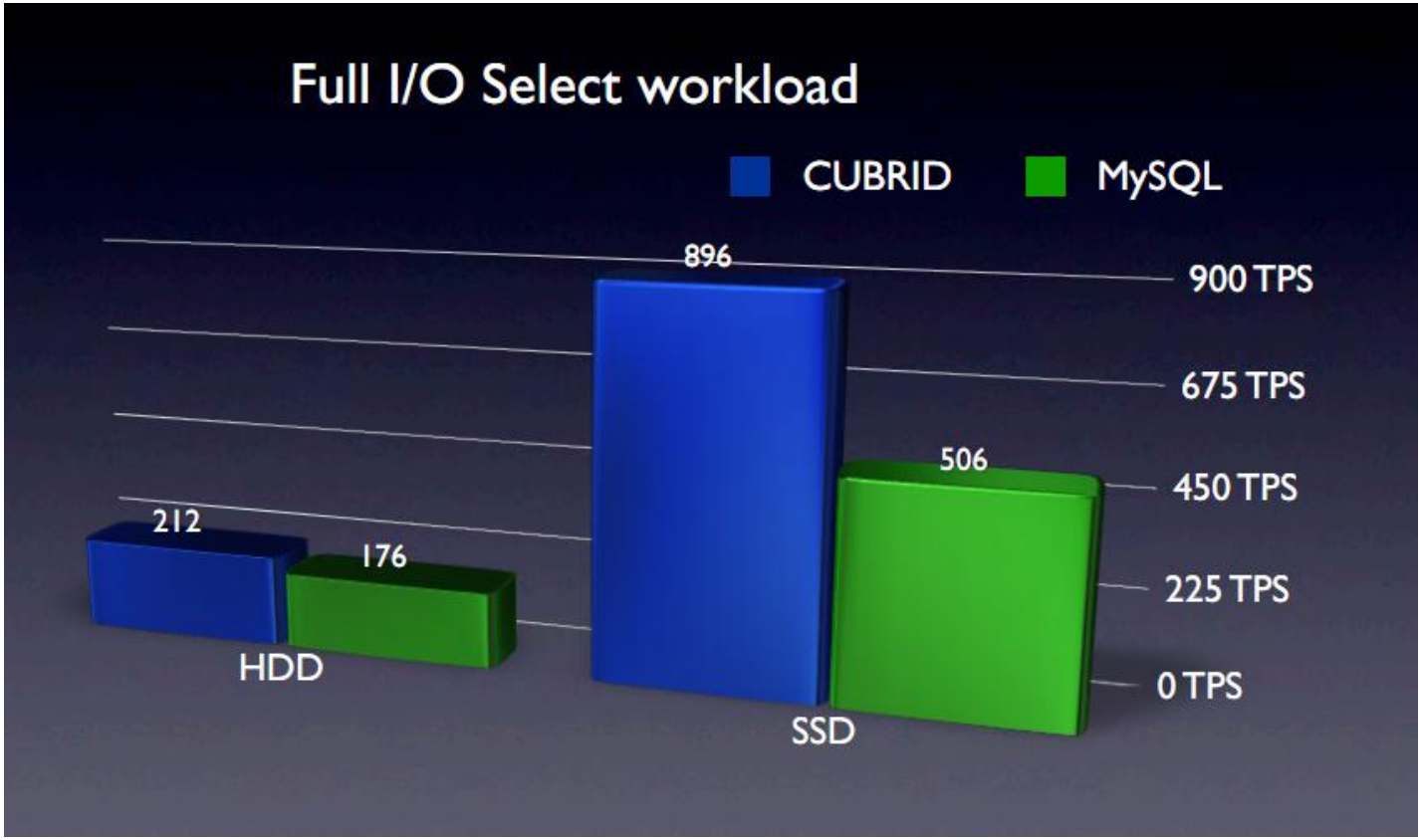
CUBRID + SSD

- SSD 사용 시 성능(TPS) 향상
  - 일반 HDD vs Solid State Drive
  - 파일럿 시험 수행 결과 (INSERT & SELECT)



CUBRID + SSD

- 성능 향상
  - CUBRID 4.2배 / MySQL 2.8배



CUBRID Cluster Project

- Overview
  - The CUBRID Cluster is a spin-off project of this CUBRID DBMS.
  - The objective is to provide the linear scalability without modifying the application.
  - This will be possible through location/failure/replication/relocation transparency by global schema.
  - CUBRID Cluster will support the single database view while providing the multi access point to database.
- 오픈 소스 프로젝트
  - http://www.cubrid.org/cluster
  - http://sf.net/projects/cubridcluster/ sourceforge

Thank you.